VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra kybernetiky a biomedicínského inženýrství

Návrh a realizace lékařského modulu pro informační systém očního centra

Design and Implementation of Doctor Module for Information System Eye Center

2019 Lukáš Granzer

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra kybernetiky a biomedicínského inženýrství

Zadání bakalářské práce

Student: **Lukáš Granzer**

Studijní program: B2649 Elektrotechnika

Studijní obor: 2612R041 Řídicí a informační systémy

Téma: Návrh a realizace lékařského modulu pro informační systém očního centra
Design and Implementation of Doctor Module for Information System Eye Center

Jazyk vypracování: čeština

Zásady pro vypracování:

1. Popis současného stavu informačního systému očního centra.
2. Popis možností jazyka C# pro tvorbu PC aplikací.
3. Systémy řízení báze dat - přehled prostředků.
4. Návrh lékařského modulu pro informační systém oční kliniky.
5. Implementace lékařského modulu v jazyce C# včetně komunikace s databází.
6. Testování modulu v praxi.
7. Zhodnocení výsledků práce a závěr.

Seznam doporučené odborné literatury:

[1] ROBINSON, Simon. *C#: programujeme profesionálně*. Vyd. 1. Brno: Computer Press, 2003. Programmer to programmer. ISBN 80-251-0085-5.
[2] NAGEL, Christian, Jay GLYNN a Morgan SKINNER. *Professional C# 5.0 and .NET 4.5.1*. Indianapolis, IN: John Wiley and Sons, 2014. ISBN 978-1-118-83294-3.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Jaromír Konečný, Ph.D.**

Datum zadání: 01.09.2018

Datum odevzdání: 30.04.2019

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA Fakulta elektrotechniky a informatiky 1D16 OSTRAVA

doc. Ing. Jiří Koziorek, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 30. dubna, 2019

Lukáš Granzer

Tímto bych chtěl poděkovat mému vedoucímu práce Ing. Jaromírovi Konečnému, Ph.D. za cenné rady a předané zkušenosti a všechen čas, který mi věnoval při konzultacích. Zároveň děkuji své rodině za podporu při studiu.

Abstrakt

Cílem této bakalářské práce je vytvoření desktopové aplikace, zajišťující lékařům přístup do informačního systému očního centra. Aplikace bude sloužit pro zobrazení výsledků optometrických vyšetření. Dále se zde zabývám rozborem stávajícího stavu informačního systému očního centra, datovou strukturou a samotnou realizací aplikace v jazyce C#.

Klíčová slova: barvocit, zraková ostrost, databáze vyšetření, C#, Firebird, .NET Framework

Abstract

The goal of this bachelor thesis is to create desktop application for doctors, that provides access to the information system eye center. The application will be used for visualization of optometric results. Furthermore, I analyzed current status of information system, data structure and at the end, you can find description of the realization application in C# language.

Key Words: color vision of the eye, visual acuity, database of medical examinations, C#, Firebird, .NET Framework

Obsah

Seznam použitých zkratek a symbolů

API	–	Application Programming Interface
CLR	–	Common Language Runtime
FNO	–	Fakultní nemocnice v Ostravě
GPL	–	General Public License
IP	–	Internet protocol
IPL	–	Interbase Public License
IS	–	Informační systém
MSIL	–	Microsoft Intermediate Language
PDF	–	Portable Document Format
RAID	–	Redundant Array of Independent Disks
SŘBD	–	Systém řízení báze dat

Seznam obrázků

Seznam tabulek

Seznam výpisů zdrojového kódu

1 Úvod

Cílem bakalářské práce je navrhnout a realizovat desktopovou aplikaci, která umožní lékařům přístup k databázi pacientů očního centra a v přehledné formě zobrazit dostupné informace o daném pacientovi. Aplikace bude součástí jednoho velkého celku, který dohromady tvoří komplexní informační systém spravující veškeré informace o návštěvách pacientů, provedených vyšetřeních a výsledcích vyšetření. Po dokončení všech potřebných modulů má tento softwarový balík dostatečný potenciál k tomu, aby nahradil stávající způsob uchovávání informací o pacientech formou kartotéky.

V první části bakalářské práce se věnuji popisu již hotových částí informačního systému, jejich funkcionalitou a návazností na nový lékařský modul. V další části se věnuji rozboru možností vývoje aplikací v jazyce C#, především v kombinaci s frameworkem .NET. Ve zkratce zde nastiňuji způsob, jakým je možné připojovat k vyvíjeným aplikacím knihovny, které ale nejsou napsány v jazyce C#.

V následující kapitole najdete stručný přehled různých systémů řízení bází dat. Provádím zde porovnávání jednotlivých systémů z hlediska jejich náročnosti na systémové prostředky. Ve zkratce jsou zde uvedeny i některé odlišné způsoby přístupu k ukládání dat. Především se ale zaměřuji na systém Firebird, protože je nedílnou součástí vyvíjeného softwaru.

V praktické části se věnuji realizaci desktopové aplikace. Najdete zde popis implementace použitých technologií a řešení dílčích funkcionalit programu. Vzhledem k tomu, že je značná část IS již téměř hotová, je vývoj programu velkou měrou spjat s implementací hotových komponent a jejich vylepšováním. Funkcionalita nového programu byla stanovena velmi obecně, proto bylo nutné konzultovat výslednou implementaci s pracovníky očního centra, aby se vývoj programu zbytečně neodchyloval od požadovaného cíle.

1.1 Prohlášení o ochraně osobních údajů

K otestování bakalářské práce byl použit datový kompilát, který částečně obsahoval reálné informace o pacientech. Veškeré výstupy z této aplikace, které obsahují osobní informace, byly proto rozmazány. Ostatní zveřejněné údaje jsou náhodná testovací data.

2 Současný stav IS

Informační systém očního centra ve Fakultní nemocnici v Ostravě je založen na konceptu samostatných desktopových aplikací, které pracují se společnými daty z hlavní databáze na nemocničním serveru. Dále je zde jedna aplikace pro tablety s operačním systémem Android. Všechny aplikace jsou navrženy pouze pro práci v online režimu, tudíž je nelze provozovat bez funkčního spojení s databázovým serverem. Aplikace si neuchovávají žádná data na lokálních zařízeních a všechny provedené změny se okamžitě ukládají na server. Jedinou výjimkou jsou v tomto ohledu konfigurační soubory aplikací, do kterých si každá aplikace ukládá poslední hodnoty některých vstupních polí, jako je například uživatelské jméno posledního přihlášeného uživatele.

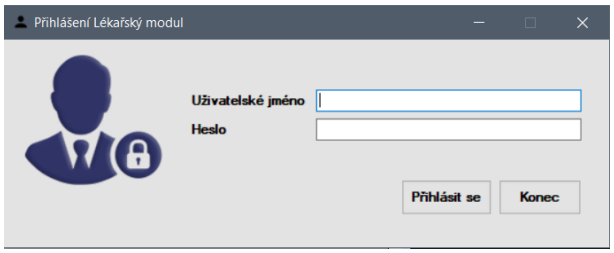
V době psaní této bakalářské práce byly hotové tyto aplikace:

- Test barvocitu – HueTest
- Test ostrosti zraku – aplikace pro tablet
- Administrátorský program – OcniAmbulanceAdmin
- Aplikace pro správu pacientů – OcniAmbulanceUser

Všechny aplikace, s výjimkou testu ostrosti zraku, jsou napsány v jazyku C#. Pouze test ostrosti zraku je napsán v programovacím jazyku Java. Přístup do informačního systému přes aplikace je chráněn pomocí autentizace uživatele.

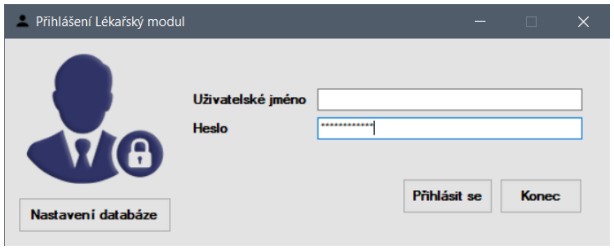
2.1 Přihlášení do IS

Přihlašovací formulář umožňuje jednoduché ověření, zda uživatel má patřičnou úroveň oprávnění pro spuštění daného programu. Při inicializaci komponenty se v konstruktoru objektu zadá požadovaná úroveň oprávnění, která je vyžadována pro spuštění daného programu. Přihlašovací údaje uživatele se porovnají s údaji v databázi a zkontroluje se, zda má uživatel dostatečné oprávnění ke spuštění aplikace.



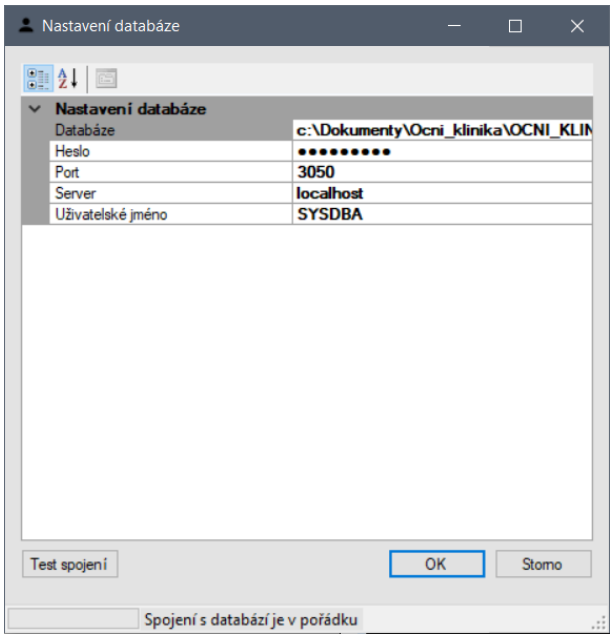
Obrázek 1: Přihlašovací okno

Další funkcí, kterou disponuje tato komponenta, je možnost měnit konfiguraci připojení k databázovému serveru. Pokud uživatel zadá do pole heslo „*showsettings*“, objeví se v dolním rohu skryté tlačítko, které umožní vstup do nastavení připojení databáze.



Obrázek 2: Skryté tlačítko pro vstup do nastavení

Zde je poté možno provést konfiguraci připojení k databázovému serveru. V případě ostrého nasazení na produkci se do vstupu *Databáze* zadá IP adresa nemocničního serveru.

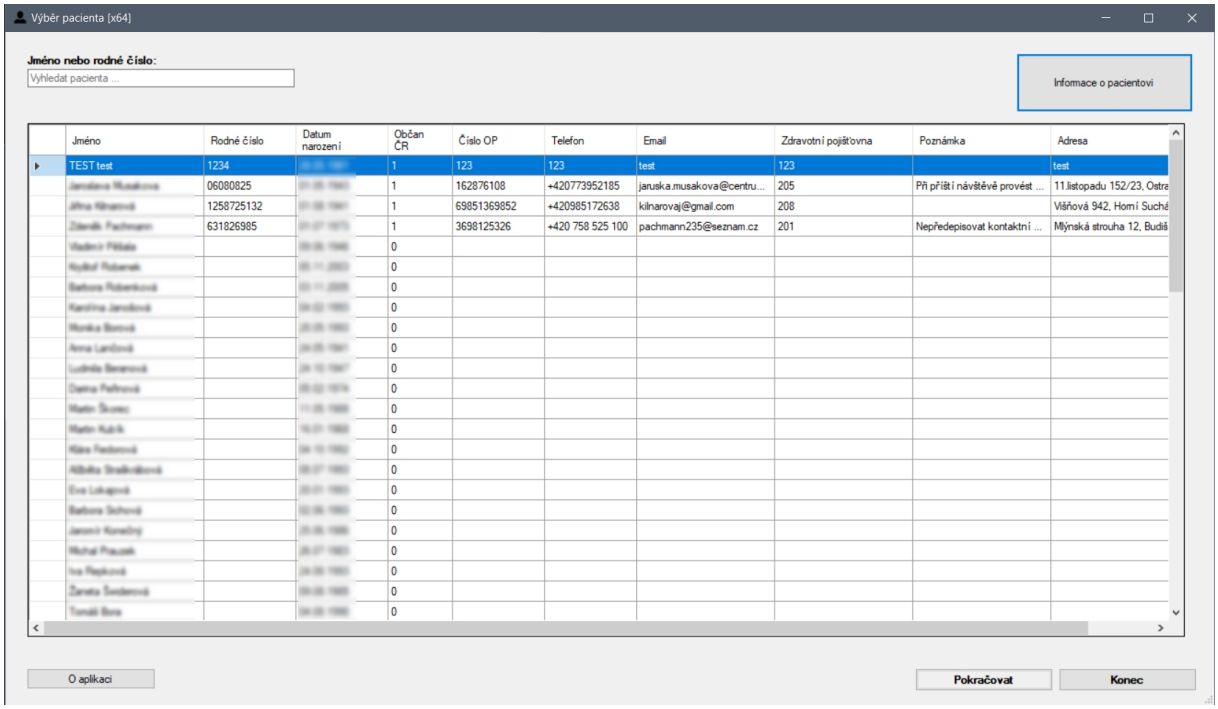


Obrázek 3: Nastavení připojení k databázovému serveru

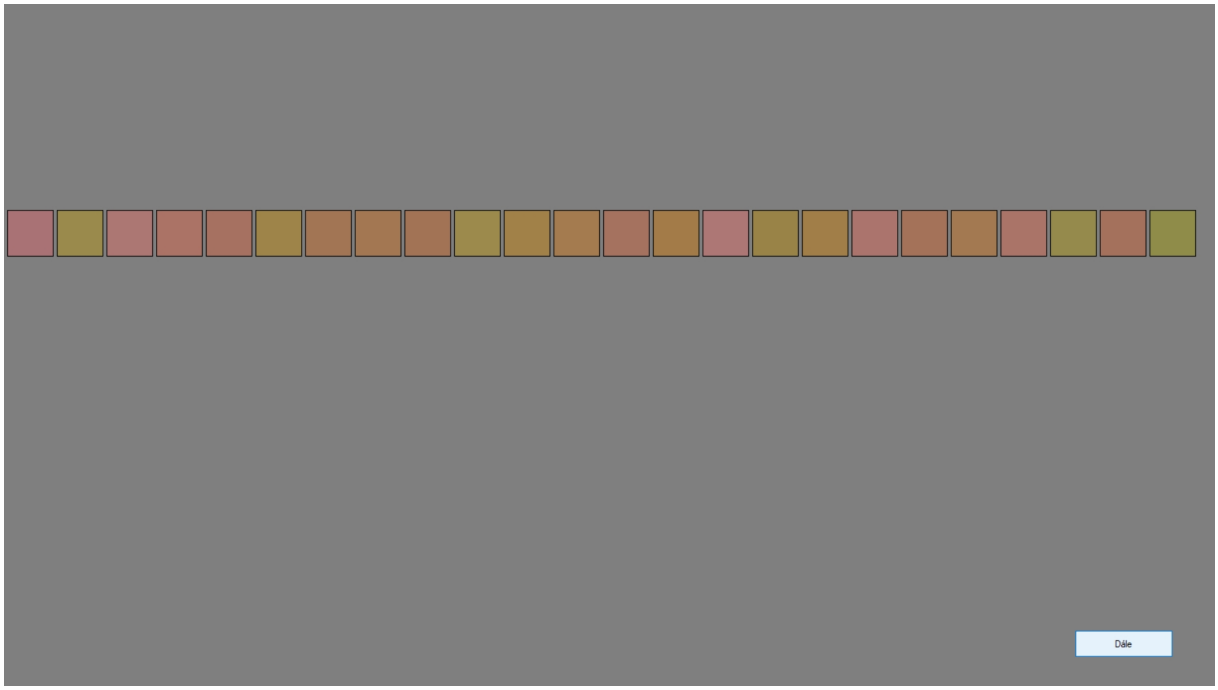
2.2 Test barvocitu

Tento program slouží k testování barvocitu pacienta. Test se provádí na stolním počítači. Zdravotní sestra nebo lékař provede v programu výběr pacienta a spustí mu vybraný typ testu. Pacient postupuje podle pokynů na obrazovce. Po dokončení testu se výsledky testu uloží na server, případně je možné provést export výsledku do formátu PDF. Po zobrazení výsledku testu se aplikace ukončí, čímž se zabrání tomu, aby pacient po dokončení testu prováděl neoprávněné zásahy do informačního systém.

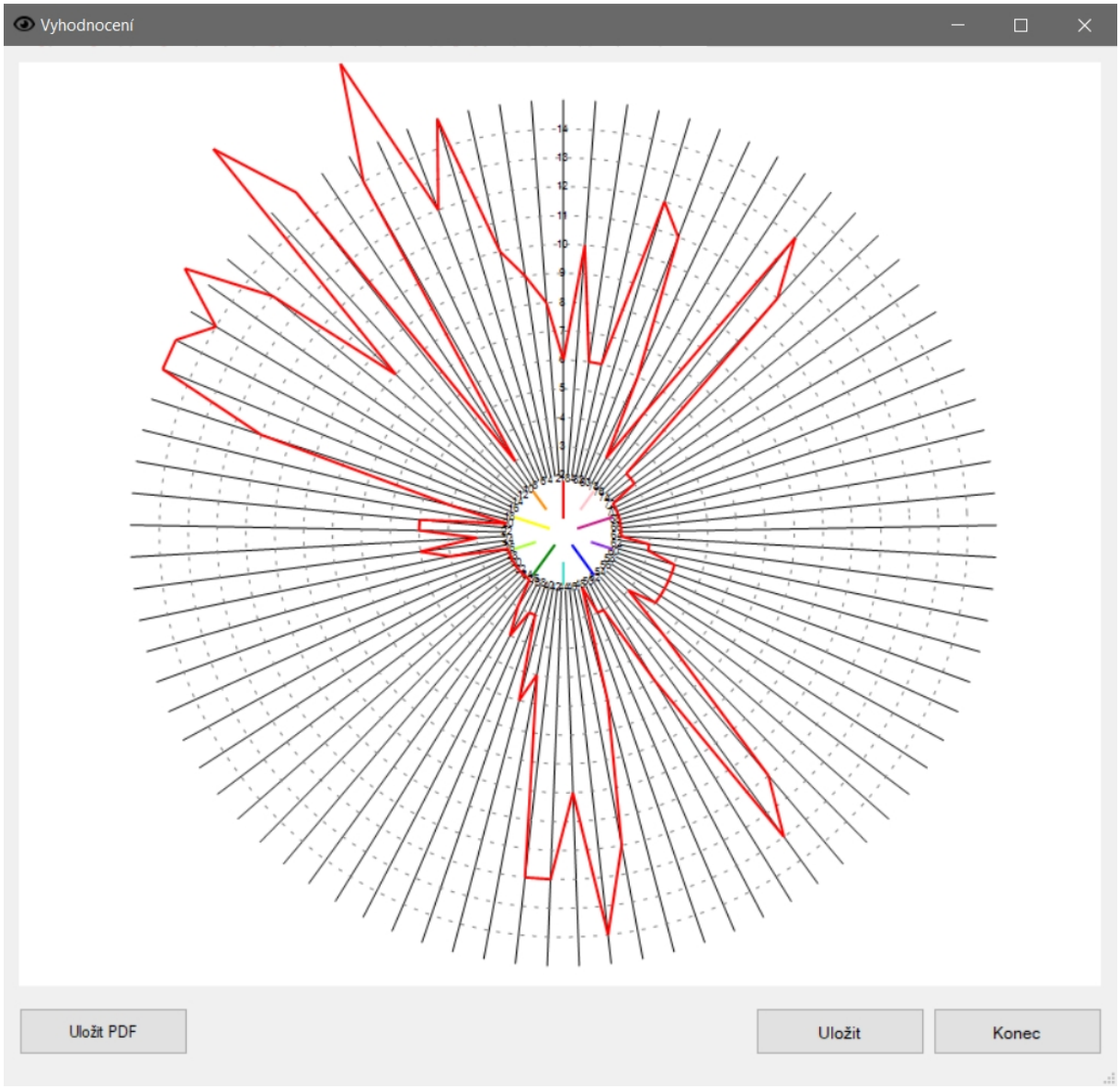
V současné době nabízí program 2 typy testů barvocitu a to Farnsworthův-Munsellův 100 test a Farnsworthův-Munsellův D15 test.



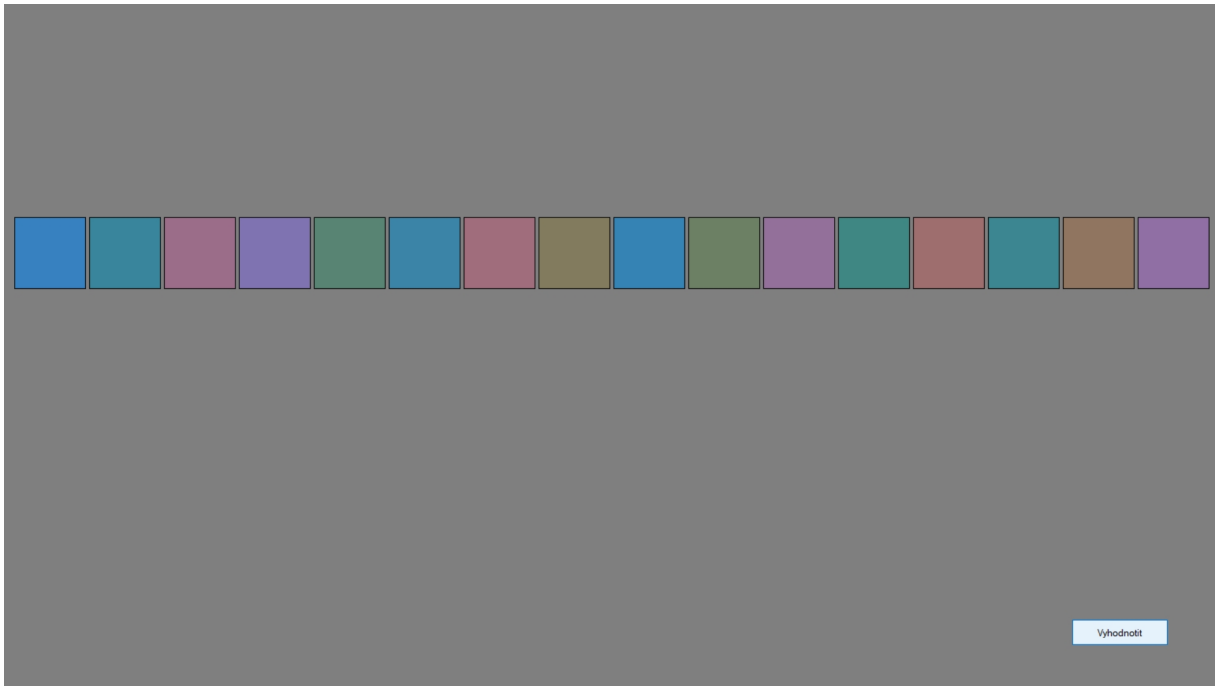
Obrázek 4: Výběr pacienta



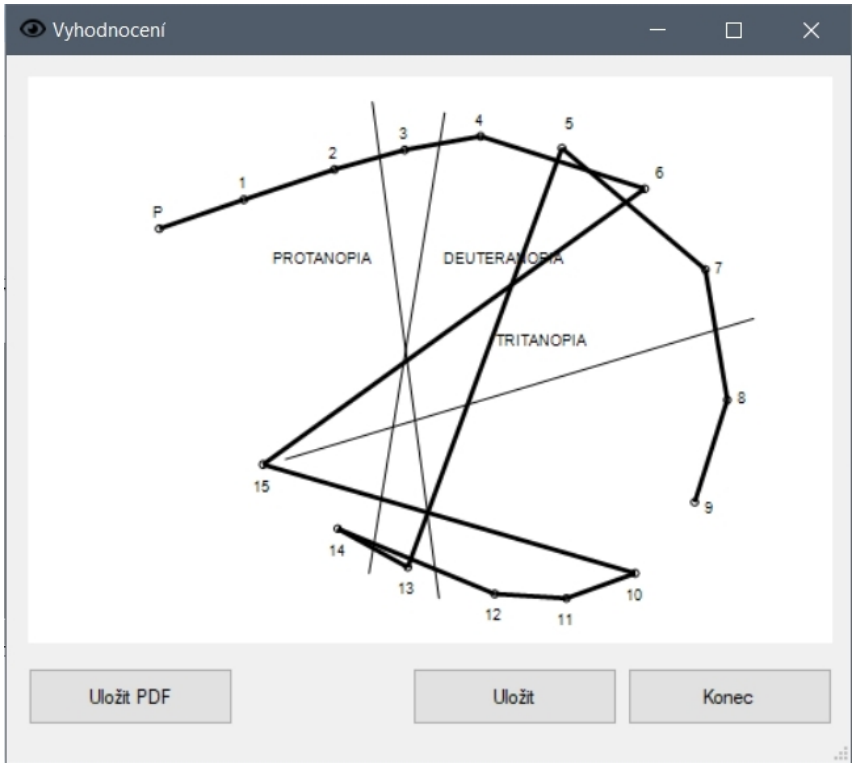
Obrázek 5: Ukázka části Farnsworth-Munsell 100 testu



Obrázek 6: Vizualizace výsledku F. - M. 100 testu



Obrázek 7: Ukázka Farnsworthova testu D-15



Obrázek 8: Vizualizace výsledku D-15 testu

Výsledek testu se uloží do databázové tabulky *HUE_EXAMINATION_RESULTS*. Kom-

pletní výsledek obsahuje v závislosti na typu testu celkem 15 nebo 85 položek, proto zde uvedu jen zkrácený výpis výsledků. Ve sloupci *COLOR_ORDER* můžeme vidět pořadí testovacích čtverců tak, jak je seřadil pacient. V případě dobrého barvocitu pacienta by se měl sloupec *COLOR_ORDER* shodovat se sloupcem *COLOR_INDEX*. Sloupec *EXAMINATION_ID* je unikátní číslo testu, podle kterého je možné dohledat záznamy konkrétního testu. Sloupec *HUE_EXAMINATION_ID* je jedinečný identifikátor každého záznamu.

HUE_EXAMINATION_ID	EXAMINATION_ID	COLOR_INDEX	COLOR_ORDER
86	3	1	2
76	3	2	1
85	3	3	5
79	3	4	6
78	3	5	4
100	3	6	7
91	3	7	3
95	3	8	51
80	3	9	52

Tabulka 1: Zkrácený databázový záznam výsledku Farnsworth-Munsell 100 testu

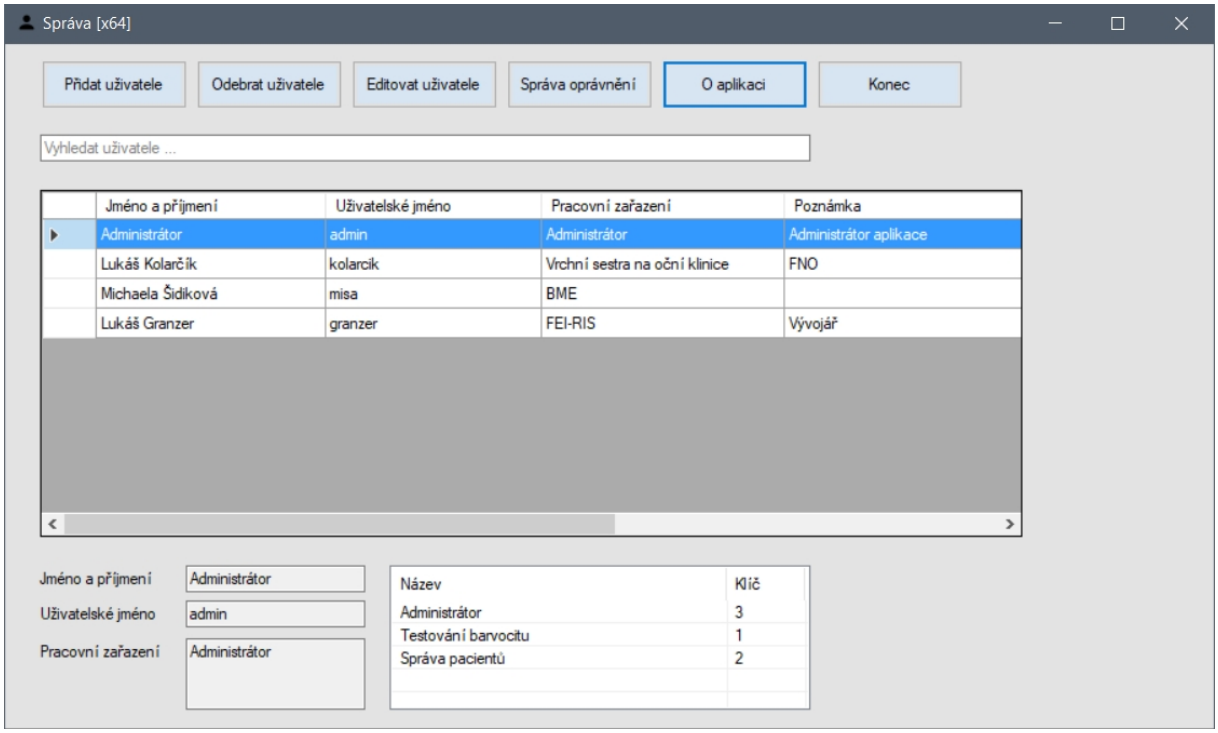
2.3 Test ostrosti zraku

Tato aplikace pro operační systém Android je prozatím jediným zdrojem dat pro testování ostrosti zraku. Podrobné informace o této aplikaci je možné najít v bakalářské práci s názvem *Návrh a realizace modulu hodnocení zrakové ostrosti pro informační systém očního centra*, jejíž autorkou je Michaela Šidiková.

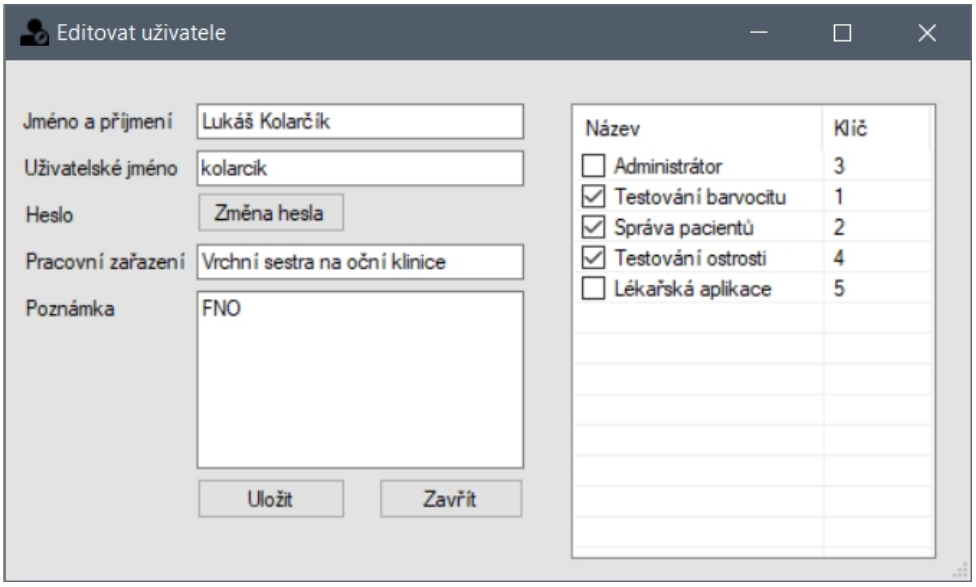
2.4 Administrátorský program

Program slouží ke správě uživatelských účtů zaměstnanců očního centra. Umožňuje vytváření a mazání zaměstnaneckých účtů, změnu přihlašovacího hesla a nastavení přístupových práv.

Na obrázku č. 9 můžeme vidět rozložení ovládacích prvků okna administrátorského programu. Uprostřed okna se nachází komponenta *dataGridView*, pomocí které můžeme vybrat konkrétního zaměstnance nebo jej vyhledat pomocí vyhledávače nad touto komponentou. Po označení zaměstnance v *dataGridView* se v dolní části okna zobrazí doplňující informace o daném zaměstnanci.

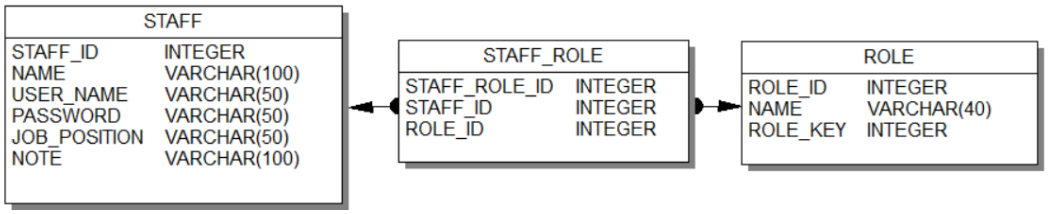


Obrázek 9: Vzhled administrátorského programu

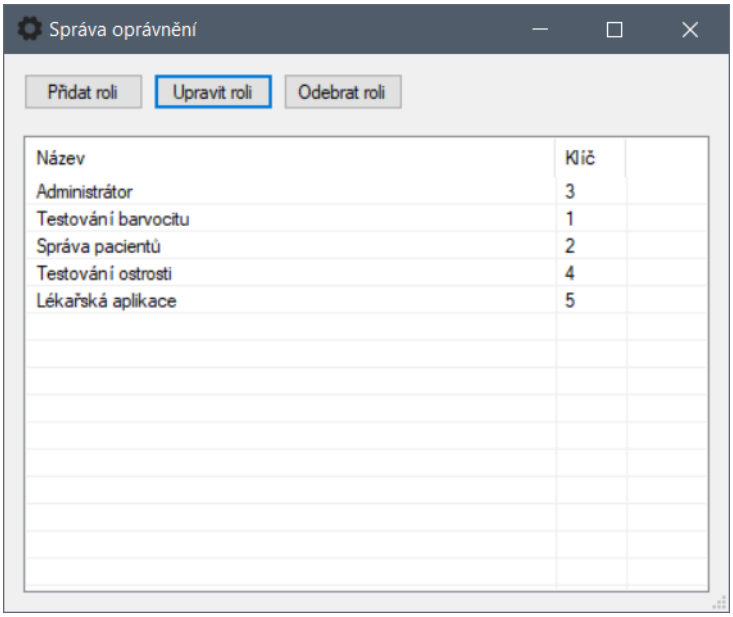


Obrázek 10: Ukázka editace uživatelského účtu

Tlačítko *Správa oprávnění* (viz obrázek č. 9) otevírá okno, ve kterém je možné vytvářet a mazat tzv. klíče k jednotlivým aplikacím. Tyto klíče jsou uloženy v databázové tabulce *ROLE*. Pomocí pomocné tabulky *STAFF_ROLE* je realizována vazba N:M mezi tabulkou zaměstnanců a tabulkou aplikačních klíčů. Při přihlašování zaměstnance do IS se provádí dotaz na databázi, který ověřuje přiřazení daného klíče aplikace k přihlašovanému zaměstnanci. Pokud není nalezeno žádné přiřazení vybraného klíče v tabulce *STAFF_ROLE*, znamená to, že zaměstnanec nemá oprávnění ke spuštění dané aplikace.



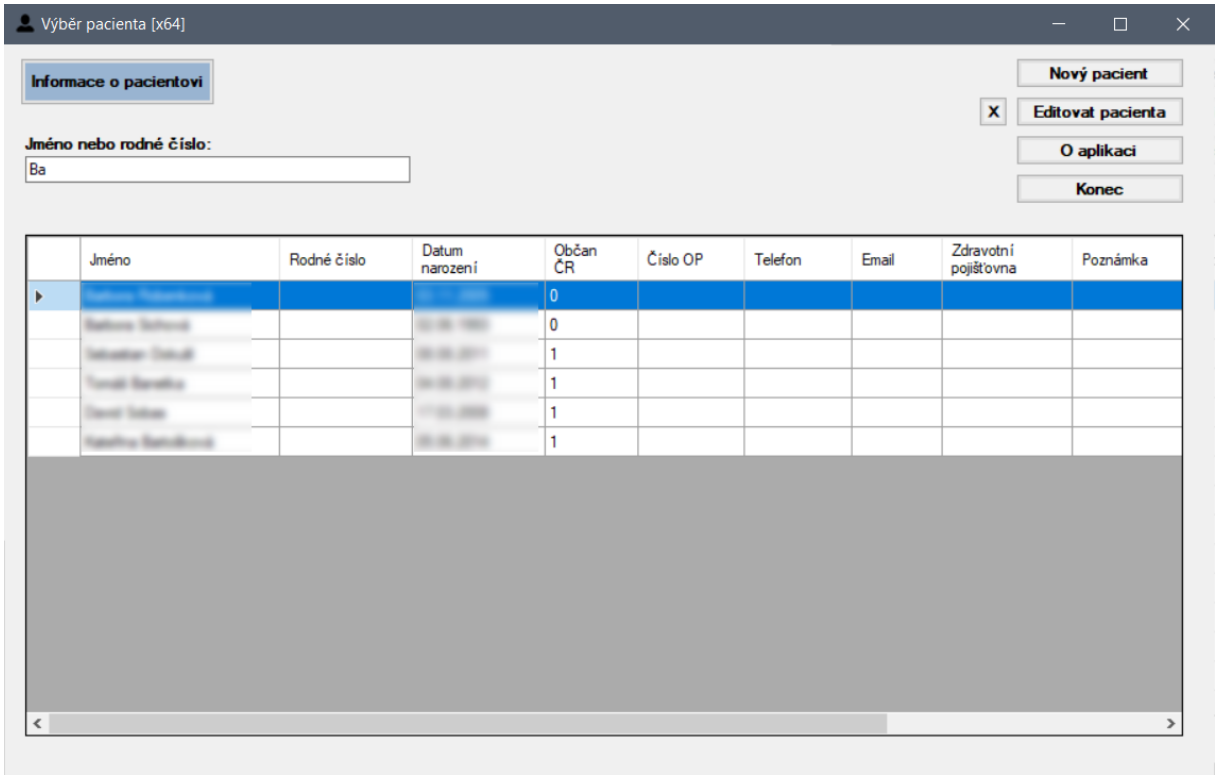
Obrázek 11: Vazba mezi tabulkou zaměstanců a tabulkou aplikačních klíčů



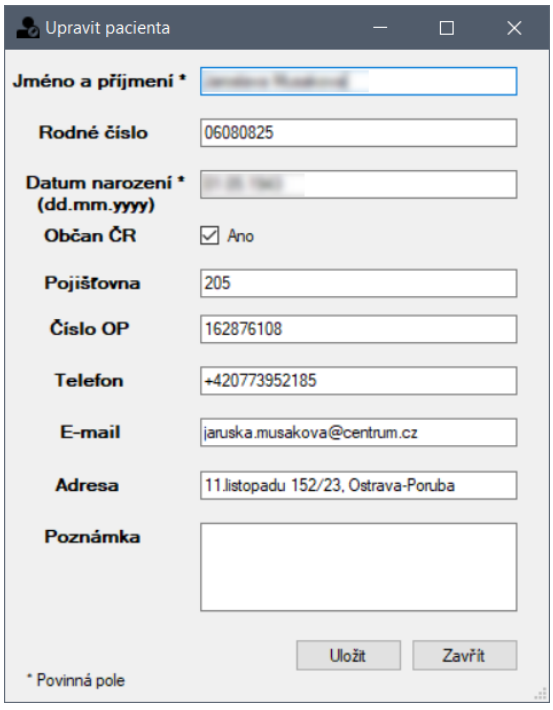
Obrázek 12: Okno pro správu oprávnění

2.5 Aplikace pro správu pacientů

Tento program je určen pro zdravotní sestru na příjmu pacientů. Program umožňuje vyhledat pacienta v databázi na základě jména, příjmení nebo rodného čísla. Po označení pacienta je možné si zobrazit historii všech vyšetření. Tento program je navíc jediným prostředkem, pomocí kterého je možné přidávat nové pacienty do databáze, eventuálně provádět změny osobních údajů pacienta.

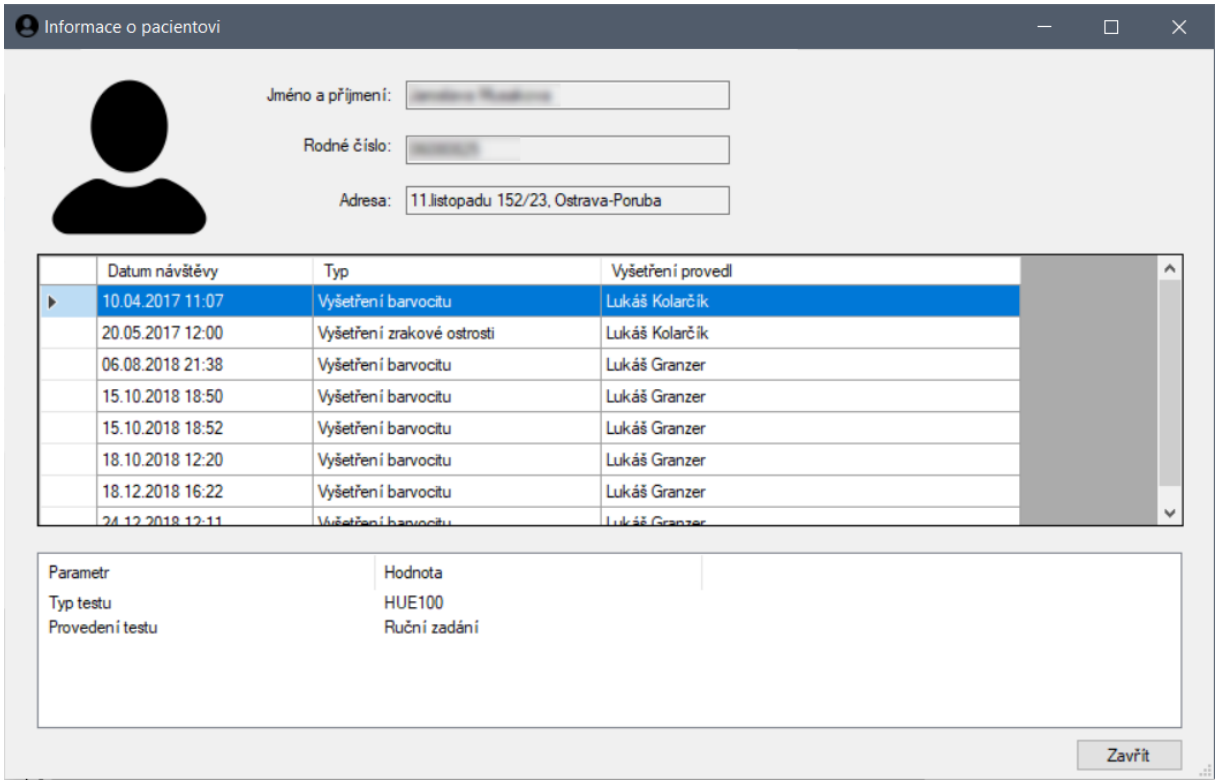


Obrázek 13: Výpis pacientů



Obrázek 14: Okno pro upravování osobních informací

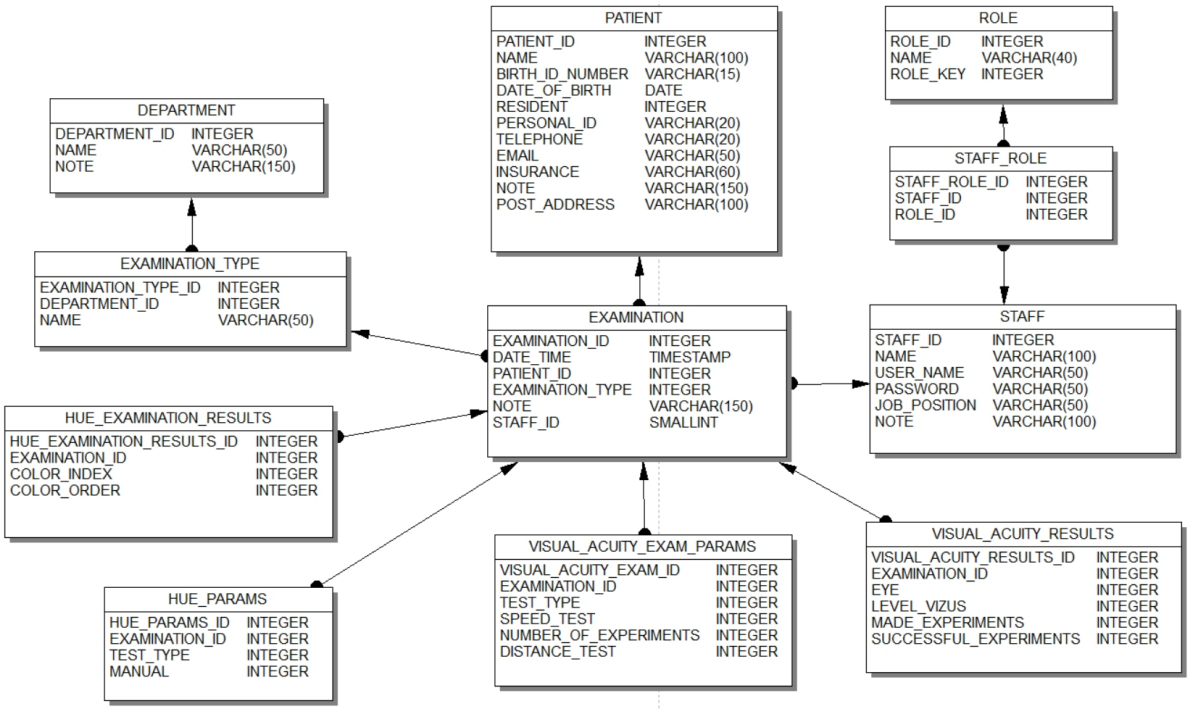
Po kliknutí na tlačítko *Informace o pacientovi* (obrázek č. 13) se otevře okno s kartou pacienta. V tomto okně se uživateli zobrazí historie všech vyšetření daného pacienta. V dolní části okna se zobrazují podrobnosti k vyšetření, které je právě označeno modrým pruhem ve výpisu historie. Tento program neumožňuje zobrazení výsledku vyšetření. Pro tyto účely bude sloužit nový program, který je předmětem této bakalářské práce.



Obrázek 15: Historie všech vyšetření

3 Datová struktura IS

Všechna data, která obsahuje IS očního centra FNO, jsou uložena v jedné relační databázi s databázovým systémem Firebird. Tabulky se zobrazením vzájemných vazeb jsou zobrazeny na obrázku č. 16. Směr šipek udává vztah mezi tabulkami, všechny vazby jsou typu 1:N.



Obrázek 16: Databázová struktura

Hlavní tabulkou databáze je tabulka provedených vyšetření *EXAMINATION*. Tato tabulka se z valné části skládá z cizích klíčů, které odkazují na další záznamy související s tímto vyšetřením. Osobní údaje pacientů jsou uloženy v tabulce *PATIENT*, přičemž primárním klíčem tabulky je identifikátor *PATIENT_ID*.

Účty personálu očního centra jsou uloženy v tabulce *STAFF*. Tato tabulka je propojená s tabulkou *ROLE*, ve které je uložen seznam všech možných úrovní oprávnění přístupu. Pomocí tabulky *STAFF_ROLE* je vytvořena vazba s kardinalitou M:N mezi tabulkou *STAFF* a *ROLE*.

V tabulkách *DEPARTMENT* a *EXAMINATION_TYPE* jsou zaznamenána jednotlivá oddělení očního centra a jednotlivé typy vyšetření, která je možné provádět na pracovištích. Podle sloupce typ vyšetření, v tabulce *EXAMINATION*, aplikace určuje, ve kterých tabulkách jsou uložena data s výsledky vyšetření. V současné době jsou naprogramovány 2 typy vyšetření, takže databáze obsahuje 2 skupiny tabulek, konkrétně *HUE_EXAMINATION_RESULTS*, *HUE_PARAMS* a *VISUAL_ACUITY_EXAM_PARAMS* a *VISUAL_ACUITY_RESULTS*, ve kterých jsou uloženy výsledky vyšetření.

4 Možnosti jazyka C# pro tvorbu PC aplikací

V roce 2002 se objevil na programátorské scéně nový, velmi silný programovací nástroj, jazyk C#. Za jeho vznikem stojí společnost Microsoft. V kombinaci s operačním prostředím .NET Framework se jedná o jednu z hlavních technologií pro vývoj aplikačního softwaru v prostředí operačního systému Windows. O kvalitě této technologie svědčí fakt, že v současné době vyšla již 7. verze tohoto jazyka (konkrétně 7.3.) a očekává se další vývoj.[2]

Při návrhu jazyka C# byl jeho zakladateli kladen důraz na tyto vlastnosti.

- Jednoduchost, víceúčelovost a objektově orientovaný programovací přístup.
- Poskytnutí podpory základním principům softwarového inženýrství, např. kontrola datových typů, hlídání hranic polí, detekce neinicializovaných proměnných nebo automatický úklid paměťového prostoru.
- Přenositelnost zdrojových kódů, obzvlášť vůči jazykům jako jsou C a C++.
- Mezinárodní podpora
- Možnost vývoje aplikací jak pro plnohodnotné operační systémy, tak pro nízkoúrovňové aplikace jako jsou vestavěné systémy.
- Úsporná práce s přiděleným procesorovým časem a pamětí.

V případě zájmu můžete plné znění specifikace tohoto jazyka najít na webových stránkách organizace *ECMA international*. [3]

Programy napsané v jazyku C# musí před spuštěním projít procesem kompilace, který je rozdělen do dvou kroků. Prvním krokem k vytvoření spustitelného kódu je překlad zdrojového kódu do jazyka MSIL (Microsoft Intermediate Language). Tento překlad se děje těsně před startem aplikace. Výhodou tohoto přístupu je možnost pružně přizpůsobit výsledný kód cílové platformě, což by při běžné kompilaci kódu a následné distribuci výsledného programu nebylo možné, protože v době kompilace častokrát neznáme hardwarovou konfiguraci cílového stroje. Navíc je tento krok optimalizován tak, že se překládá jen ta část programu, která je v tento okamžik nutná ke spuštění programu. Zbývající části programu zůstávají nepřeloženy až do chvíle, kdy dojde k jejich zavolání běžícím programem. Jedná se o tzv. překlad Just-In-Time. Byť se tento postup může zdát pomalejší oproti jednorázovému překladu celého programu, firma Microsoft vychází z předpokladu, že je vysoká pravděpodobnost, že uživatel nevyužije veškerou funkcionalitu programu. V tomto případě bude tento přístup efektivnější než jednorázový kompletní překlad.[4][1]

Druhým krokem kompilace je spuštění modulu CLR (Common Language Runtime), což je modul operačního prostředí .NET Framework. Modul překládá kód v jazyce MSIL do spustitelné podoby. Překládaný kód MSIL navíc nemusí nutně pocházet z kódu napsaného v jazyce C#, ale může se jednat o libovolné knihovny napsané v některém z jazyků, které jsou kompatibilní

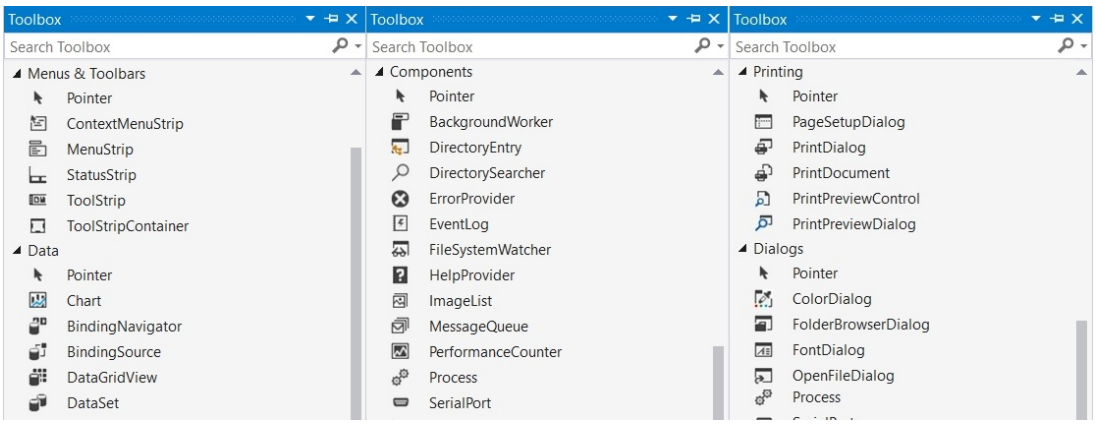
s normou MSIL, jako je např. VB.NET, J# nebo řízené C++. Tímto se vývojářům otevírají nové možnosti v propojování různých knihoven napříč programovacími jazyky. [4][1]

4.1 Technologie .NET Framework

Jedná se o rozsáhlou knihovnu, poskytující široké spektrum předpřipravených komponent, které je možné integrovat do vyvíjeného softwaru. Knihovna je primárně určena pro operační systém Windows, ale existuje i její open source alternativa MONO s více méně stejnou funkcionalitou. S pomocí obsahu knihovny .NET Frameworku je možné vytvořit tyto typy aplikací:[5]

- Konzolové aplikace
- Windows Forms aplikace
- Windows Presentation Foundation - WPF
- ASP.NET
- Windows services
- Windows Communication Foundation - WCF
- Windows Workflow Foundation - WF

Výše uvedený seznam představuje základní rozdělení frameworku. Každá část obsahuje specifické funkce, které jsou typické pro daný typ aplikace. K vytvoření lékařského modulu byla použita především skupina komponent Windows Forms. Windows Forms poskytuje kvalitní platformu pro tvorbu formulářových aplikací. Obsahuje různé druhy dialogových oken, formulářové prvky (textová pole, tlačítka, multimediální prvky, menu) i objekty pro práci s operačním systémem (logy, řízení procesů, přístupy k souborovému systému).[6]



Obrázek 17: Ukázka části palety Windows Forms

Pro snadný vývoj aplikací s využitím těchto komponent existuje velmi kvalitní podpora ve vývojářském nástroji Microsoft Visual Studio, které obsahuje grafický designer aplikací.

5 Systémy řízení báze dat

K uchovávání a práci s rozsáhlými daty slouží v informatice různé typy systémů řízení bází dat, dále jen SŘBD. Úlohou těchto systémů je řízení operací s uloženými daty a řízení přístupu k těmto datům. SŘBD zajišťuje klientům komfortní přístup k hledaným informacím a přebírá za něj zodpovědnost nad organizováním přístupu k těmto datům. Při souběhu více požadavků musí tyto systémy řešit synchronizaci změn a v rámci možností i kontrolovat vkládaná data. Častokrát jsou uložená data ještě chráněna pomocí přístupových práv, které umožňují omezit některým uživatelům provádění konkrétních operací, např. mazání nebo i čtení určitých dat. Při ukládání nových dat zajišťují indexaci nového záznamu vzhledem k ostatním datům, aby se urychlilo jeho pozdější vyhledání. Během provádění změn dat dokáží tyto systémy zabránit ztrátám nebo poškození dat při nenadálých pádech klientských aplikací.

Volbu typu SŘBD ovlivňuje mnoho faktorů, například:

- Struktura a různorodost uložených dat
- Množství uložených dat
- Počet současně obsluhovaných klientů
- Požadovaná dostupnost dat vs. integrita dat

Z hlediska struktury uložených dat a způsobu ukládání můžeme rozdělit SŘBD a jejich úložiště do několika základních skupin.

5.1 Relační databáze

Základem těchto databází jsou tabulky s pevně danou strukturou. Každý záznam je identifikován pomocí jedinečného primárního klíče, který označuje konkrétní řádek v tabulce. Sloupec tabulky obsahuje vždy pouze data jednoho datového typu a má svůj vlastní název. V ideálním případě obsahuje sloupec jen jednu, dále nedělitelnou informaci. Vztahy mezi daty jsou vyjádřeny pomocí odkazování se na primární klíče v dalších tabulkách.

Nevýhodou tohoto způsobu ukládání dat je jeho nízká flexibilita, co do schopnosti pojmout záznamy s rozdílnou povahou vlastností. Pevná struktura tabulek nám totiž přímo neumožňuje ukládat rozdílné počty vlastností pro jednotlivé řádky záznamu. Relační databáze kladou vysoký důraz na konzistenci dat na úkor dostupnosti. Prováděné změny se projeví až po potvrzení transakce, takže přerušení spojení nebo pád aplikace nezpůsobí poškození právě modifikovaných záznamů.

5.1.1 MySQL

MySQL je jeden z nejpoužívanějších open-sourcových systémů pro řízení bází dat. Systém je šířen pod dvojí licencí, a to GPL nebo placenou variantou. Lze jej nasadit jak na operačním

systému MS Windows, tak v systémech s operačním systémem Linux. Jedná se o systém s relačním modelem zpracování dat. Tento SŘBD je častokrát nasazován na serverech v kombinaci s webovým serverem Apach a technologií PHP, tzv. LAMP(Linux-Apache-MySQL-PHP).

5.1.2 PostgreSQL

PostgreSQL je stabilní a bezpečný SŘBD, který se velmi osvědčil v rozsáhlých aplikacích. Z hlediska licence je dovoleno jej volně používat a modifikovat, a to i pro komerční účely. Systém je kvalitně zdokumentovaný a má vynikající uživatelskou podporu. O jeho vývoj a podporu se stará rozsáhlá komunita nezávislých vývojářů, která je otevřena novým členům.

K tomuto systému existuje nesčetné množství doplňků a modifikací, které rozšiřují už tak velmi širokou paletu funkcionalit tohoto systému. Jako příklad uvedu rozšíření PostGIS, které umožňuje pracovat s geografickými daty. Systém je velmi vhodný pro aplikace, kde se očekává velký objem ukládaných dat a byl nasazen např. pro službu Skype nebo OpenStreetMap. [8, 9]

5.1.3 Firebird

Na základě uvolněných zdrojových kódů SŘBD InterBase od firmy Borland vznikl projekt Firebird. O vývoj se stará mezinárodní tým vývojářů, který se sdružuje pod neziskovou organizací a má podporu několika spřízněných firem. Firebird je šířen pod licencí InterBase Public License. Zjednodušeně řečeno, je možné systém volně používat ve vlastních komerčních projektech. Dále je možné systém libovolně upravovat pro vlastní potřebu. V případě šíření vlastní modifikované verze, musí být tato upravená verze šířena pod licencí IPL.[10, 11]

Systém je možné provozovat pod operačním systémem Windows i Linuxových distribucích. Pro optimální výkon se doporučuje provozovat diskové úložiště dat v RAID 0, 3 nebo 5. Další možnou optimalizací je rozdělit datový soubor databáze na více souborů a rozdistribuovat je mezi více diskových polí. Jádro systému Firebird je známé svými nízkými systémovými nároky. Na druhou stranu si zachovalo velmi rychlou odezvu a umožňuje spravovat databáze až do velikosti 30 TB. Databázový server je možné provozovat v několika konfiguracích (architekturách):[10, 11]

- Embedded
  Jedná se o jednu knihovnu, která obsahuje jak serverovou, tak klientskou část, potřebnou pro komunikaci s databází. Používá se jako strukturované úložiště dat pro program, do kterého je implementována. Může sice poskytovat data i dalším programům v rámci jednoho počítače, ale neumožňuje síťovou komunikaci. Navíc zde neexistuje možnost omezení přístupových práv k určitým datům. Program zpřístupňuje všechna data, ke kterým má v rámci daného uživatelského účtu na úrovni operačního systému přístup.

- SuperServer
  V režimu SuperServeru je spuštěn na serveru jeden proces s jednou vyrovnávací pamětí

(tzv. cache). Výhodou této sestavy je dobrá dostupnost dat pro všechny klienty, protože sdílená cache v sobě uchovává značnou zásobu dat. Tato konfigurace neumožňuje využít vícejádrové procesory, proto se hodí pouze pro malé aplikace lokálního rázu s malým množstvím dat.

- Classic
  Architektura Classic vytváří pro každé spojení jeden samostatný proces s vlastní vyrovnávací pamětí. Nevýhodou je menší efektivita využití cache, protože obsahuje pouze data, s kterými se pracovalo v rámci tohoto spojení. Na druhou stranu, kritická chyba jednoho spojení nijak neovlivní ostatní připojené klienty. Problém nastává v případě velkého počtu současně připojených klientů, který způsobí vytvoření stejného počtu běžících serverových procesů, čímž dojde k značnému zvýšení zátěže operačního systému, který musí zvládnout obsluhu všech procesů, což se v samém důsledku projeví jako zpomalení chodu. V praxi se doporučuje provoz do počtu 100 aktivních klientů.

- SuperClassic
  Odstraňuje hlavní nedostatek architektury Classic, a to velký počet těžko obsluhovatelných procesů. Na serveru běží pouze jeden proces a příchozí spojení vytváří pouze nové vlákno, v tomto procesu. Díky tomu je možné synchronizovat vyrovnávací paměti přímo v operační paměti RAM, bez potřeby I/O operací. V samém důsledku nám tak odpadne značná část rutinních systémových operací, jinak potřebných pro obsluhu velkého počtu separátních procesů. [11, 12]

5.2 Porovnání nároků na systémové prostředky

Na závěr bych chtěl porovnat nároky jednotlivých systémů řízení bází dat na hardwarové prostředky serveru. Ve sloupci *Velikost jádra* najdete velikost diskového prostoru, který je potřeba alokovat pro samotný databázový server bez dat. V případě MySQL serveru se jedná pouze o orientační hodnotu, velikost jádra závisí na mnoha faktorech. Sloupec *RAM pro server* udává velikost paměti RAM, která je potřebná pro běh samotného serveru. V této hodnotě už může být částečně započten i objem dat potřebných pro obsluhu klienta. Sloupec *RAM pro připojení* udává velikost paměti RAM, která se alokuje pro obsluhu každého nově připojeného klienta.

Můžeme vidět, že z hlediska systémových nároků je použitý SŘBD Firebird o jeden řád úspornější co do nároků na diskový prostor. I pro samotný chod jádra jsou jeho paměťové nároky zanedbatelné vzhledem k porovnávaným produktům.

Databázový systém	Velikost jádra	RAM pro server	RAM pro připojení
MySQL	cca 300 MB	576 MB [1]	256 kB
PostgreSQL	700 MB	8 GB [2]	10 MB
Firebird - superserver	12 MB	3 MB	115 kB
Firebird - classic	12 MB	0 kB	2 MB

Tabulka 2: Srovnání nároků na systémové prostředky

5.3 NoSQL databáze

Not Only SQL databáze fungují na principu shlukování souvisejících informací do jednoho společného záznamu. Tato uskupení nemusí mít nutně stejnou datovou strukturu, takže je možné záznamům libovolně přiřazovat i nestrukturované vlastnosti a doplňující informace. Mnohé systému umožňují pružně volit mezi důrazem na konzistenci dat a rychlostí odezvy. [7]

5.3.1 MongoDB

Databázový systém založený na modelu dokumentového úložiště. Filozofií tohoto systému je uchovávat související informace pohromadě v minimálním počtu databázových objektů. Databázi je možné provozovat formou distribuovaného systému a provádět balancování serverové zátěže. Součástí vnitřního chodu je i mechanizmus duplikace záznamů, čímž se omezuje dopad havárie hardwaru na uložená data. Systém lze použít i jako souborový systém. Databázový systém našel své uplatnění u společností jako jsou Twitter, Ebay, Cisco, UPS.

5.4 Objektově orientované databáze

Vychází z konceptu objektově orientovaného programování, kdy se spolu s daty ukládají i postupy, jak s těmito daty pracovat. V praxi se tento přístup většinou realizuje oklikou, pomocí technologie ORM (object-relation mapping). Pomocí softwaru se vytvoří komunikační kanál, který na jedné straně pracuje se standardní relační databází, ale na druhé straně komunikace vrací data formou nadefinovaných objektů.

[1] Výchozí nastavení pro 150 současných spojení s klienty.
[2] Doporučená velikost paměti RAM pro chod systému.

6 Praktická část

Desktopová část informačního systému pro FNO je vyvíjena v programu Microsoft Visual Studio pro platformu Microsoft .NET Framework 4.5., který najdeme v operačních systémech Windows 7, 8, 10 a Windows Vista.

6.1 Požadovaná funkcionalita lékařského modulu

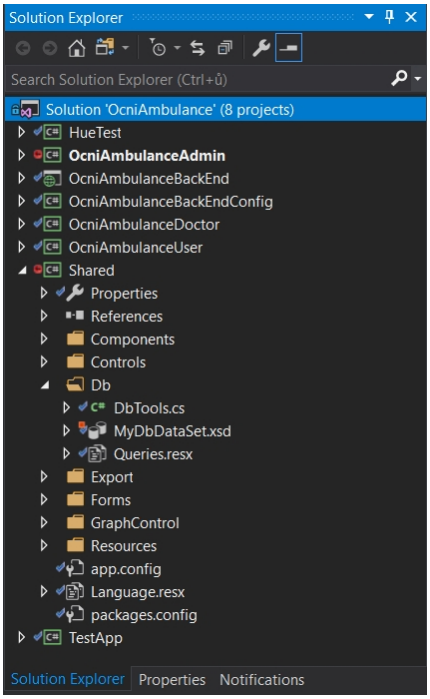
Na základě rozboru zadání nového lékařského modulu byla identifikována tato funkcionalita:

- Přihlašování uživatele
- Vyhledání pacienta podle jména, příjmení nebo rodného čísla
- Zobrazení historie vyšetření
- Vizualizace všech typů výsledků vyšetření
- Export výsledků vyšetření ve formátu PDF
- Tisk výsledků

6.2 Struktura projektu

Během vývoje IS, na kterém se postupně podílela řada studentů a studentek, vznikla tato struktura projektu. Knihovna *Shared* obsahuje všechny třídy a funkce, které je možné libovolně implementovat v jiných aplikacích. Jakékoliv modifikaci této knihovny musí předcházet důkladná analýza kódu ostatních aplikací, aby se neporušila funkcionalita již hotových aplikací. Jednou částí této knihovny je definice datasetu *MyDbDataSet.xsd*, který objektově definuje strukturu databáze, na niž je projekt připojen. Dále je zde mimo jiné soubor *Queries.resx* definující SQL dotazy, pomocí kterých aplikace získává data z databáze.

Jednotlivé moduly IS jsou uloženy v separátních složkách. V současné době se zde nachází modul pro testování barvocitu (*HueTest*), modul pro správu uživatelů (*OcniAmbulanceAdmin*), modul pro správu pacientů (*OcniAmbulanceUser*) a nový modul *OcniAmbulanceDoctor*. Moduly *TestApp* a *OcniAmbulanceBackendConfig* jsou pouze pomocné programy určené pro vývojáře.



Obrázek 18: Struktura projektu

6.3 Autentizace uživatele

Při spuštění aplikace se nejprve provede ověření přístupových práv uživatele. Toto ověření je realizováno pomocí hotové knihovny *Shared.Forms.LoginForm*. Vytvořené instanci objektu této třídy je předána číselná hodnota požadované úrovně oprávnění (tzv. role key), kterou vyžaduje daná aplikace. Dále je potřeba předat textový řetězec s loginem posledního přihlášeného uživatele, databázový konektor a informace o konfiguraci databáze.

Uživateli se zobrazí přihlašovací okno, do kterého zadá své jméno a heslo. Zadané údaje se porovnají s databází zaměstnanců a tabulkou přístupových práv. Pokud má uživatel patřičné oprávnění, vrátí objekt *login* hodnotu *DialogResult.OK* a dojde k otevření hlavního okna aplikace.

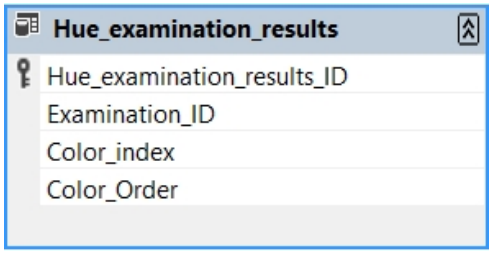
Přihlašovací formulář umožňuje měnit nastavení připojení k databázovému serveru, ale tato funkcionalita je pro běžného uživatele skryta. Více informací k této funkcionalitě najdete v kapitole 2.1.

6.4 Zpracování dat z databáze

Protože je celý program napsaný v objektově orientovaném jazyce, bylo by chybou nevyužít možnosti objektového přístupu k databázovým datům. Tímto odpadne potřeba složité datové konverze při čtení a zápisu dat do databáze. K tomuto účelu byla použita knihovna *DataSet* z frameworku .NET. Pomocí této knihovny je možné si s využitím grafického editoru nadefinovat strukturu třídy, která svými vnitřními proměnnými kopíruje strukturu vybrané databázové

tabulky nebo pouze určitého výběru dat. Vyčtená data z databáze můžeme následně uchovávat v programu pomocí instancí této třídy, čímž dojde k zachování logické struktury zpracovávaných informací.

Výhody uvedené technologie si můžeme prakticky ukázat na tomto příkladu. Program potřebuje načíst a zpracovat data o výsledku testu barvocitu. Víme, že každý záznam bude obsahovat jedinečný číselný identifikátor, číslo vyšetření, číslo barevného odstínu a hodnotu, kterou mu přiřadil diagnostikovaný pacient.



Obrázek 19: Definice objektu datasetu v prostředí Visual Studio

Pomocí průvodce ve Visual Studiu byla nadefinována výše uvedená struktura datasetu. Po sestavení SQL dotazu předáme tento textový řetězec funkci z knihovny *DBTools*. Tato knihovna zprostředkovává napojení na databázový server Firebird.

```
select * from Hue_examination_results
where Examination_ID=5 ORDER BY COLOR_ORDER;
```

Výpis 1: Načtení výsledků vyšetření číslo 5

Příchozí data z databázového serveru jsou zpracována touto knihovnou a načtena do instance třídy datasetu, takže výstupem databázového dotazu je objekt, který obsahuje všechna data, která jsme získali aktuálním dotazem z databáze.

```
MyDataSet ds = new MyDataSet();
// Sestavení databázového dotazu
string query = string.Format(Queries.SELECT_EXAMINATION_HUE_RESULTS_BY_ID,
                             row.Examination_ID);
// Provedení dotazu a uložení výsledku do datasetu
DbTools.SelectQuery(Program.DatabaseConnection,
                    ds.Hue_examination_results,
                    query);
// Kopírování prvního záznamu z pole výsledků
MyDataSet.Hue_examination_resultsRow firstResult
                                        = ds.Hue_examination_results[0];
```

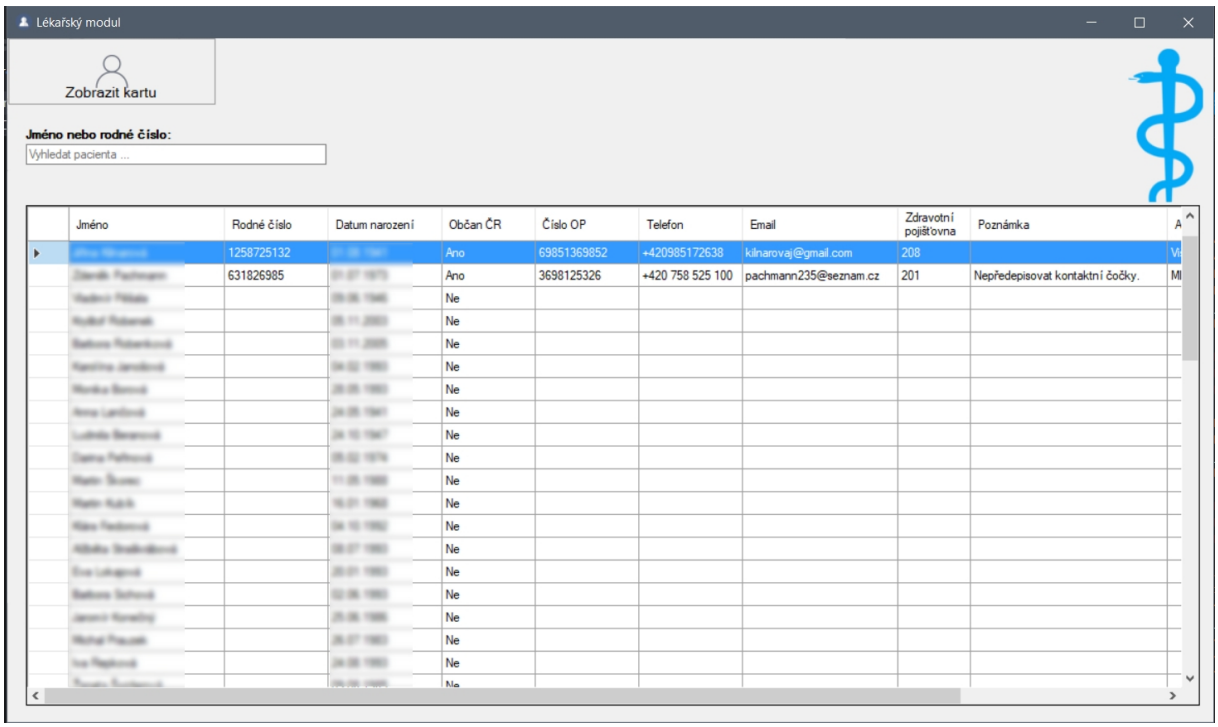
```
// Ukázka přístupu k jednotlivým údajům v záznamu
firstResult.Hue_examination_results_ID;
firstResult.Examination_ID;
firstResult.Color_index;
firstResult.Color_Order;
```

Výpis 2: Ukázka objektového přístupu k databázovým datům

Tímto způsobem je realizována veškerá práce s databázovými daty.

6.5 Hlavní okno

Hlavní okno programu zobrazuje formou tabulky seznam všech pacientů očního centra FNO. Výpis je možné seřadit podle libovolného sloupce tabulky kliknutím na záhlaví sloupce. Tabulka pacientů spolu s textovým vstupem nad tabulkou tvoří dohromady jednu komponentu, která byla naprogramována některým z mých předchůdců a nachází se ve sdílené knihovně *Shared.Controls*. K její implementaci bylo potřeba pouze navázat spojení s databázovým serverem a předat komponentě objekt s databázovým konektorem. Komponenta vrací informaci o aktivním řádku, který byl vybrán uživatelem. Kliknutí na tlačítko *Zobrazit kartu* otevře další okno s kartou vybraného pacienta.



Obrázek 20: Hlavní okno aplikace

6.5.1 Hledání pacienta v databázi

Textové pole s popiskem *Jméno nebo rodné číslo* na obrázku č. 20 umožňuje vyhledat konkrétního pacienta v databázi. Při testování této funkcionality jsem odhalil drobný nedostatek ve vyhledávacím algoritmu, který ale značně znepříjemňoval používání programu.

Pokud uživatel zadal do vyhledávacího pole jméno nebo příjmení pacienta, ale nezadal počáteční písmeno velkým písmenem, program častokrát zcela nelogicky nezobrazil žádného pacienta. Příčinou byl způsob, jakým byl napsaný vyhledávací SQL dotaz.

```
select * from Patient
where Name like '%{0}%' or Birth_id_number like '%{0}%'
```

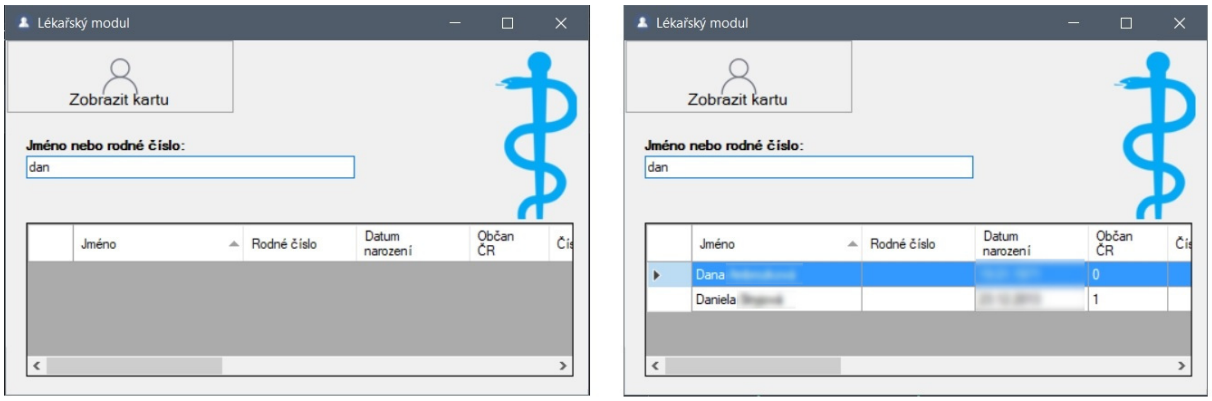
Výpis 3: Původní vyhledávací SQL dotaz

Namísto řetězce „{0}“ se doplní hledané slovo zadané uživatelem. Znaky % před a za tímto řetězcem indikují, že se před nebo za hledaným výrazem můžou nacházet ještě nějaké nespecifikované znaky, tímto máme vytvořený vyhledávací vzor. Klíčové slovo *LIKE* znamená, že hledáme záznam, který odpovídá našemu vyhledávacímu vzoru. Problémem ale bylo, že operátor *LIKE* rozlišuje mezi malými a velkými písmeny. Proto dotaz vyhodnocoval řetezce „Novák“ a „novák“ jako zcela odlišná slova.

```
select * from Patient
where lower(Name) like lower('%{0}%') or Birth_id_number like '%{0}%'
```

Výpis 4: Vylepšený vyhledávací SQL dotaz

Řešení, které by potlačilo toto nežádoucí chování funkce *LIKE*, spočívala v převedení hledaného slova i databázového záznamu na malá písmena pomocí funkce *LOWER*. Tím je zajištěna necitlivost vyhledávacího SQL dotazu na rozdíly mezi velkými a malými písmeny.

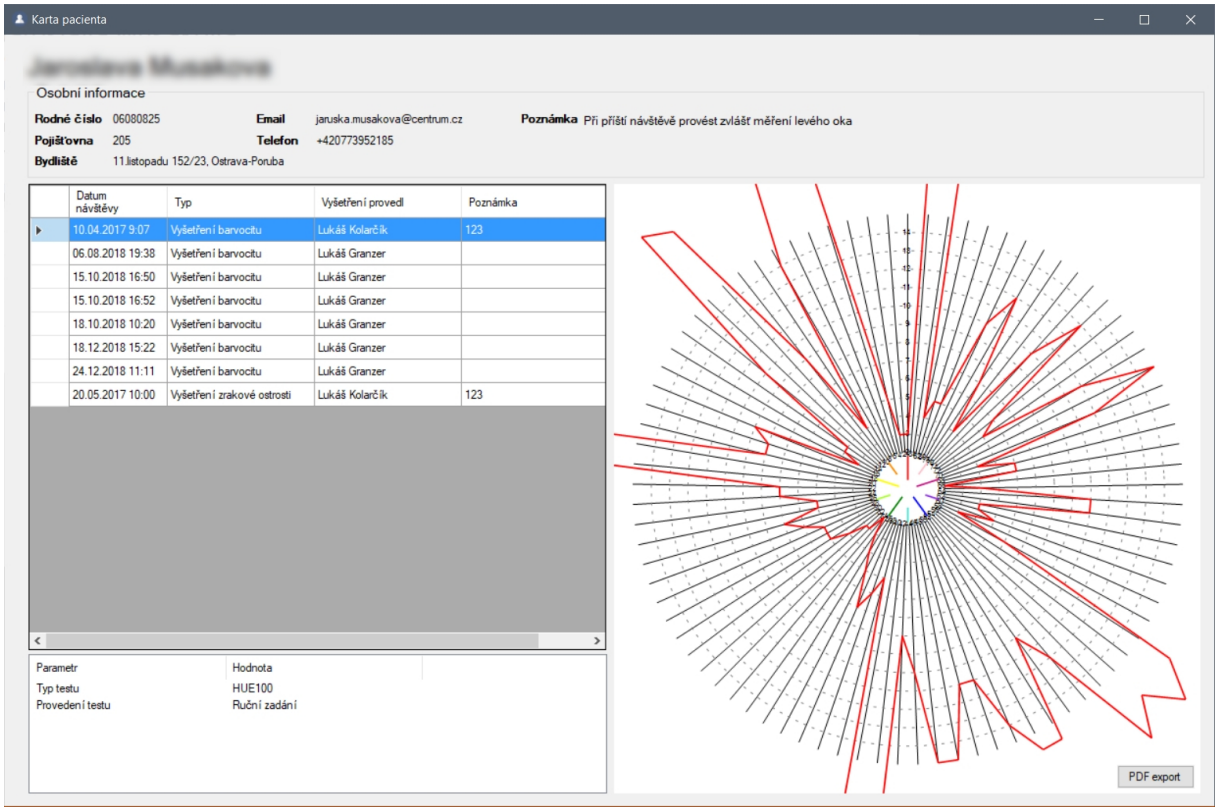


Před úpravou Po úpravě

Obrázek 21: Výsledek vyhledávání

6.6 Karta pacienta

Před otevřením okna s kartou pacienta se doplní do horní části okna všechny osobní informace pacienta, jako je jméno, příjmení, bydliště atd. Tyto informace se v tuto chvíli nenačítají z databáze, ale předávají se formou kopie objektu třídy *MyDataSet.PatientRow* z okna, ve kterém uživatel vyhledával pacienta. Následuje načítání historie návštěv pacienta, které trvá ze všech kroků při otevírání karty nejdéle. Na základě pacientova unikátního čísla se z databázových tabulek *EXAMINATION, STAFF* a *EXAMINATION_TYPE* vyberou příslušná data a načtou se do objektu třídy *MyDataSet.PatienDetailInfo*. Výsledná sestava dat se doplní do tabulkové komponenty *dataGridView*, která leží v levé prostřední části okna. Dokončení kopírování dat do *dataGridView* vyvolá další vyhledávací dotaz na databázi, který vrátí data o výsledcích prvního vyšetření v tabulce. Výsledek vyšetření se pomocí příslušné funkce zobrazí v grafické podobě v pravé polovině okna. Doplňující informace k vyšetření se zobrazí v seznamu pod historií vyšetření. Teprve v tuto chvíli se uživateli zobrazí vyplněné okno s kartou pacienta na obrazovce.



Obrázek 22: Karta pacienta

Otevřené okno s kartou pacienta má automaticky nastavený tzv. focus na komponentu s historií vyšetření. Díky tomu je možné bez jakéhokoliv dalšího kliknutí myší procházet historii vyšetření pomocí klávesových šipek. Při každé změně aktivního řádku v tabulce historie se z databáze načte výsledek vybraného vyšetření a překreslí se graf s vizualizací.

6.6.1 Vizualizace

K vizualizaci výsledku vyšetření jsem použil grafickou komponentu ze sdílené knihovny projektu *Shared.GraphControl.Graph*. Komponenta slouží k vykreslování obrázků ve vektorové podobě. Umožňuje přibližování libovolné části vykreslené grafiky a export obrázku v rastrové podobě. Pro vizualizaci všech typů výsledků vyšetření v okně lékařského modulu bylo potřeba doprogramovat knihovnu pro vykreslení výsledku vyšetření ostrosti zraku. Knihovna pro vizualizaci barvocitu již byla hotová.

VISUAL_ACUITY_RESULTS_ID	EXAMINATION_ID	EYE	LEVEL_VIZUS	MADE_EXPERIMENTS	SUCCESSFUL_EXPERIMENTS
215	108	0	1	7	7
217	108	0	2	7	7
219	108	0	3	5	4
221	108	0	4	5	4
223	108	0	5	5	4
224	108	0	6	6	5
225	108	0	7	7	4
216	108	1	1	6	5
218	108	1	2	5	4
220	108	1	3	4	3
222	108	1	4	2	0

Obrázek 23: Databázový záznam výsledku vyšetření ostrosti zraku

Pro vizualizaci výsledku tohoto typu vyšetření jsem zvolil výpis formou tabulky. Tabulka je vykreslena pomocí skládání primitivních grafických objektů, jako jsou čára, obdélník a textový řetězec. Největším úskalím bylo zajištění korektního vykreslení tabulky, při odlišných počtech provedených vyšetření pro levé a pravé oko.

	Levé oko			Pravé oko		
	pokusů	správně	úspěšnost	pokusů	správně	úspěšnost
10 / 100	7	7	100%	6	5	83%
10 / 66	7	7	100%	5	4	80%
10 / 40	5	4	80%	4	3	75%
10 / 33	5	4	80%	2	0	0%
10 / 20	5	4	80%	-	-	... %
10 / 15	6	5	83%	-	-	... %
10 / 10	7	4	57%	-	-	... %
Celkem			**82 %**			**59 %**

Obrázek 24: Vizualizace výsledku vyšetření ostrosti zraku

6.6.2 Export do PDF

Součástí lékařského programu je i možnost exportu výsledků vyšetření do formátu PDF. Kliknutím na tlačítko *PDF export* (viz. Obrázek 22.) se spustí generování PDF výstupu aktuálně zobrazeného vyšetření. Pro formátování dokumentu je použita tabulková sazba, která umožňuje vytvoření poměrně přehledného dokumentu s minimem programového kódu. Výpis obsahuje základní informace o pacientovi a vizualizaci výsledku vyšetření.

Obrázek výsledku vyšetření je rastrovou kopií vektorového obrázku, který se zobrazuje v okně karty pacienta. Při převodu obrázku na rastrovou grafiku se objevil problém s nedostatečnou kvalitou rastrového obrázku, pokud byl obrázek vyrenderován v rozměru 350 × 200 px, což je rozměr prostoru pro vykreslení obrázku do tabulky. Řešením bylo vyrenderovat rastrový obrázek ve formátu bitmapy o dvojnásobném počtu pixelů, a následně provést zmenšení obrázku na 50 % původní velikosti pomocí funkce *ScalePercent* z knihovny *iTextSharp.text.Image*. Tímto postupem bylo dosaženo zvýšení rozlišení rastrového obrázku při zachování stejného rozměru.

6.7 Tisk

Dolní pravý roh karty pacienta obsahuje mimo jiné i tlačítko Tisk. Realizace tisku není v prostředí .NET Framework zcela dořešená. Framework sice obsahuje komponentu tiskového formuláře, která umožňuje výběr tiskárny, počtu kopií dokumentu a výběr rozsahu tištěných stránek, ale tím veškerá funkcionalita končí. Vytvoření tiskové úlohy zde není nijak programově vyřešeno.

Jednou z možností vytvoření tiskové úlohy je použití komerční knihovny Spire. Demoverze této knihovny je zdarma, ale má omezený počet stran jednoho dokumentu na max. 10 stran. Cena za plnou verzi je 599 dolarů.

Alternativním řešením je použití API některého z programů pro prohlížení PDF souborů (Adobe Reader, Foxit Reader, SumatraPDF, atd.). Nevýhodou tohoto řešení je závislost lékařského modulu na programech třetí strany. I přes tuto nevýhodu bylo nakonec toto řešení implementováno do softwaru lékařského modulu. Musíme předpokládat, že se na cílovém zařízení bude nacházet alespoň jeden z podporovaných programů.

Kliknutí na tlačítko Tisk otevře dialogové okno s výběrem cílové tiskárny a počtu kopií dokumentu. Před tiskem proběhne na pozadí aplikace export výsledků do PDF souboru. Tento soubor se uloží na disk. Po potvrzení úspěšného exportu se pomocí přístupu do příkazové řádky spustí některý z programů na prohlížení PDF souborů (o který program půjde, závisí na nastavení operačního systému) a předá se mu formou parametru cesta k souboru, který se má vytisknout. Protože touto cestou není možné předat informaci o cílové tiskárně, je potřeba si vypomoci jinou cestou. Pomocí systémové knihovny *winspool.drv* se provede změna nastavení výchozí tiskárny podle výběru uživatele, a až poté se provede spuštění tisku.

Pokud se nepodaří provést libovolnou operaci v průběhu přípravy tisku, objeví se dialogové okno s informací o vzniklé chybě a ukončí se příprava tisku.

6.8 Modularizace karty pacienta

V průběhu vývoje programu mi byl vedoucím práce navržen nový koncept aplikace, který spočíval ve vytvoření modulární aplikace. Hlavní myšlenkou byla možnost přidávat do programu nové druhy vyšetření, bez potřeby kompilace celého programu. Pokud by si nemocnice vyžádala naprogramování nového druhu vyšetření, teoreticky by stačilo novou funkcionalitu dodat pouze formou dll souboru, který by se vložil do určité složky programu. Po startu lékařského programu by si hlavní program načetl všechny dostupné doplňky. Nebylo by potřeba pokaždé přeinstalovávat lékařský program.

Převést toto řešení do praxe se ale ukázalo v tomto stádiu vývoje jako nerealizovatelné. Současná struktura, založená na sdílené knihovně, není navržená na takovouto architekturu aplikace. Tím, že se značná část kódu odkazuje právě na sdílenou knihovnu *Shared.dll*, není pořádně možné provádět zásadní rozšíření funkcionality programu bez potřeby modifikace této knihovny. Další překážkou je neexistence C# rozhraní (interface) pro již hotové testy barvocitu a ostrosti zraku. Použití jednotlivých testů vyžaduje z programátorského hlediska volání metod s různými názvy a návaznostmi, což znemožňuje zavedení univerzálního přístupu ke všem testům. Pokud bychom chtěli vytvořit jednotné rozhraní pro všechny testy, znamenalo by to značné změny již hotových programů.

7 Testování aplikace

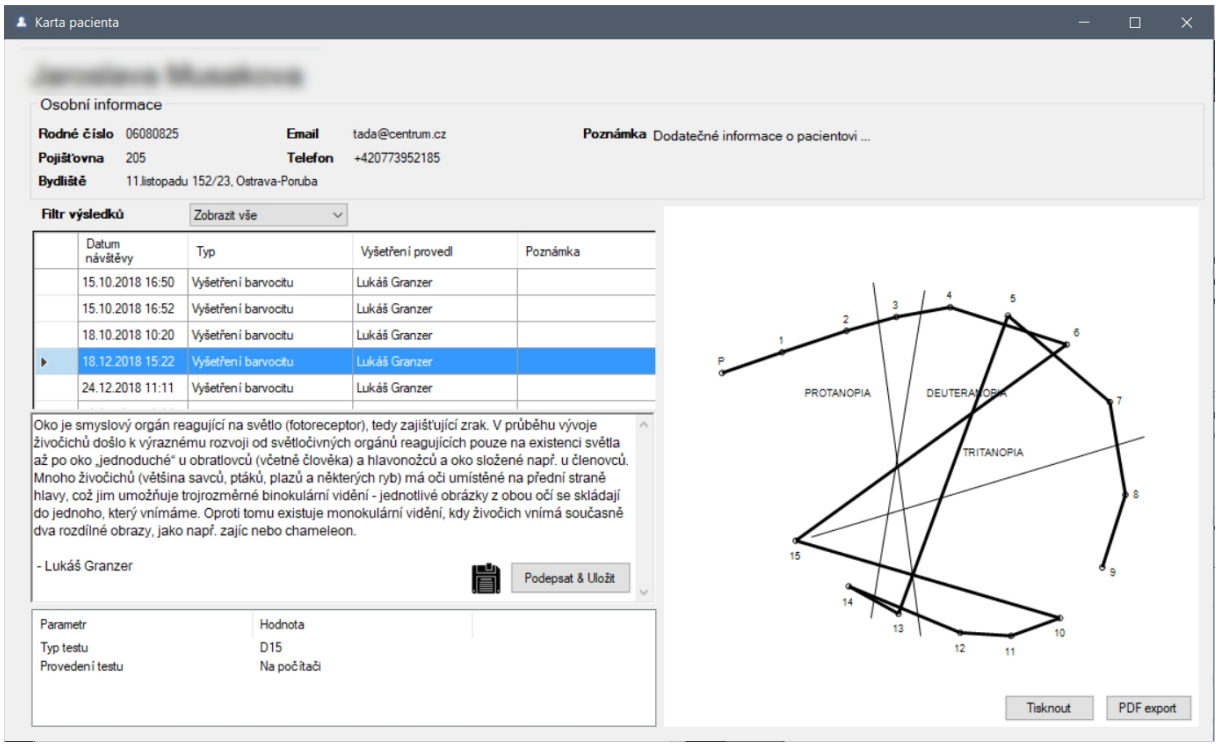
Při první konzultaci na pracovišti FNO nedošlo k samotnému otestování aplikace, ale byla provedena ukázka funkcionalit nového programu. Během konzultace se zaměstnanci očního centra mi bylo navrženo několik dalších možností, jak rozšířit funkcionalitu programu, aby více odpovídal potřebám očního centra.

7.1 Filtrace výsledků vyšetření

S ohledem na možné budoucí rozšíření programu o další typy testů by stávající chronologický výpis výsledků vyšetření nemusel být úplně přehledný. Při konzultaci byl proto vznesen požadavek na přidání možnosti filtrovat výsledky podle typu vyšetření.

7.2 Popis výsledku vyšetření

V předvedené verzi programu chyběla možnost přidávat lékařský popis výsledku vyšetření. K výsledku vyšetření bylo možné přidat krátkou poznámku, a to pouze hned po dokončení testu. Chyběla zde možnost připsat k výsledku stanovenou diagnózu. K přidání takovéto funkcionality bylo potřeba několik úprav grafiky uživatelského okna, exportu PDF dokumentů i příslušné databázové tabulky.



Obrázek 25: Nová grafika karty pacienta

Do databázové tabulky *EXAMINATION* byl přidán nový sloupec o datovém typu BLOB, který bude sloužit k uložení popisu diagnózy. Do levé poloviny okna karty pacienta byla přidána formulářová komponenta RichTextBox, pomocí níž je možné pracovat s víceřádkovým textem. V pravém dolním rohu této komponenty se nachází tlačítko *Podepsat & Uložit* a ikona signalizující stav editace textu (Uloženo/Editováno). Ve chvíli, kdy uživatel změní obsah diagnózy, se změní výše zmíněná ikona na obrázek tužky a program zaznamená změnu textu. Pokud by uživatel omylem klikl na jiné měření, zobrazí se dialogové okno s dotazem: „Uložit provedené změny v lékařské zprávě? “ a tlačítky Ano/Ne. Tím je zabráněno nechtěné ztrátě právě psané diagnózy. Po dopsání diagnózy stiskne uživatel tlačítko pro podepsání a uložení zprávy. Program načte z databáze jméno aktuálně přihlášeného lékaře a doplní jeho jméno na konec lékařské zprávy, následně se provede zápis textu do databáze a změní se ikona vedle tlačítka pro ukládání.

7.3 Další úpravy

Výše zmiňovaný text diagnózy se nově tiskne pod obrázek výsledku vyšetření. Grafika všech exportovaných PDF souborů prošla na základě podnětů pracovníků FNO radikální změnou. Bylo kompletně změněno rozložení údajů o pacientovi, přidáno logo nemocnice a text diagnózy. Ukázku těchto dokumentů je možné najít v příloze této bakalářské práce.

S přibývající funkcionalitou karty pacienta se prodloužil čas potřebný pro načtení všech dat do okna karty pacienta. Na mém počítači trvá načtení pacientovy karty o 13 záznamech okolo 4 sekund. Aby se zabránilo několikanásobnému otevření pacientovy karty v domnění, že program nic nedělá, přidal jsem do programu mechanizmus, který tomu zabrání. Po kliknutí na tlačítko *Zobrazit kartu*, se změní tomuto tlačítku popisek na „Načítám ..“ a tlačítko se dočasně zablokuje pro další stisk. Zároveň se změní kurzor na animované kolečko, které indikuje, že program pracuje. Nově otevřená karta pacienta provádí ještě krátce po svém zobrazení některé dodatečné operace s daty, které se projevují nepěkným problikáváním různých částí okna. Abych tomuto jevu zabránil, skryl jsem nově otevřené okno pomocí průhlednosti až do doby, kdy jsou dokončeny všechny potřebné operace.

8 Závěr

Cílem bakalářské práce bylo vyvinout desktopovou aplikaci, která bude začleněna do informačního systému očního centra ve Fakultní nemocnici v Ostravě. Program musí obsahovat veškerou funkcionalitu, kterou bude lékař potřebovat ke zpracování výsledků měření z oční ambulance. Vytyčeného cíle se podařilo úspěšně dosáhnout. Nově vytvořený program se podařilo dovést do plně funkčního stavu a splnit většinu požadavků zaměstnanců očního centra.

Na základě analýzy současného stavu softwaru pro informační systém mohu konstatovat, že ze strukturálního hlediska se jedná o solidně navržený systém. Software je napsán přehledně a je možné jej udržitelným způsobem dále rozšiřovat podle požadavků zákazníka. Otázkou zůstává provozovatelnost systému v dlouhodobém časovém horizontu z hlediska nástupu nových operačních systémů. Programy jsou nyní spustitelné na všech stanicích s operačním systémem Windows 7 a novějších. V jistém slova smyslu by se dala snížit závislost informačního systému na operačním systému koncových zařízení, pokud by byl informační systém napsán formou webové aplikace.

Použitý systém řízení bází dat Firebird se ukázal jako plně adekvátní volba pro tento informační systém. Nemá velké nároky na systémové prostředky a je plně dostačující pro očekávaný objem zpracovávaných dat. Vývoj lékařského modulu bude v nejbližší době nejspíše ještě pokračovat nad rámec této práce. Bylo by dobré rozšířit lékařský program o analytické nástroje, které by umožnily pokročilé prohledávaní výsledků vyšetření pro účely lékařských studií a vyhodnocování vlivu nasazené léčby na pacienta. Dále je možné rozšířit vizualizaci výsledků vyšetření barvocitu o analytické zpracování naměřených dat, které by provedlo klasifikaci dané oční vady.

Tato práce mi pomohla rozšířit si obzor hned v několika směrech. Naučil jsem se pracovat s .NET frameworkem. Seznámil jsem se s databázovým systémem Firebird, který byl pro mne novinkou a okrajově pronikl i do problematiky očních vyšetření.

Literatura

[1] ROBINSON, Simon. *C#: programujeme profesionálně.* Brno: Computer Press, 2003. Programmer to programmer. ISBN 80-251-0085-5.

[2] The history of C#. *Microsoft Docs: .NET* [online]. [cit. 2019-01-21]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/csharp/whats-new/csharp-version-history

[3] C# Language Specification: Standart ECMA. In: *ECMA international* [online]. Prosinec 2017 [cit. 2019-01-21]. Dostupné z: http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-334.pdf

[4] PETRUSHA, Ron. Přehled modelu Common Language Runtime (CLR). *Microsoft Docs: .NET* [online]. [cit. 2019-01-25]. Dostupné z: https://docs.microsoft.com/cs-cz/dotnet/standard/clr

[5] *MONO* [online]. [cit. 2019-02-04]. Dostupné z: https://www.mono-project.com/

[6] Overview of the .NET Framework. *Microsoft Docs* [online]. 30.3.2017 [cit. 2019-02-04]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/framework/get-started/overview

[7] KUBICA, Tomáš. *NoSQL: vaše jednodušší, levnější a škálovatelnější databáze.* Cloud svět [online]. [cit. 2019-02-07]. Dostupné z: http://www.cloudsvet.cz/?p=243

[8] *PostgreSQL* [online]. Benešov: Pavel Sněhule, 2019 [cit. 2019-03-01]. Dostupné z: https://postgres.cz/wiki/PostgreSQL

[9] Database. *OpenStreetMap Wiki* [online]. 8.10.2018 [cit. 2019-03-01]. Dostupné z: https://wiki.openstreetmap.org/wiki/Database

[10] *Firebird: The true open source database* [online]. Maitland, c2000-2018 [cit. 2019-03-08]. Dostupné z: https://firebirdsql.org/

[11] CÍSAŘ, Pavel. *InterBase/FireBird: podrobná příručka : tvorba, programování a správa databází.* Brno: Computer Press, 2003. ISBN 80-722-6946-1.

[12] Firebird SuperServer, ClassicServer or SuperClassic?. In: *INTItec* [online]. c2018 [cit. 2019-03-08]. Dostupné z: http://www.intitec.com/varios/Firebird-SuperServer-ClassicServer-SuperClassic.pdf

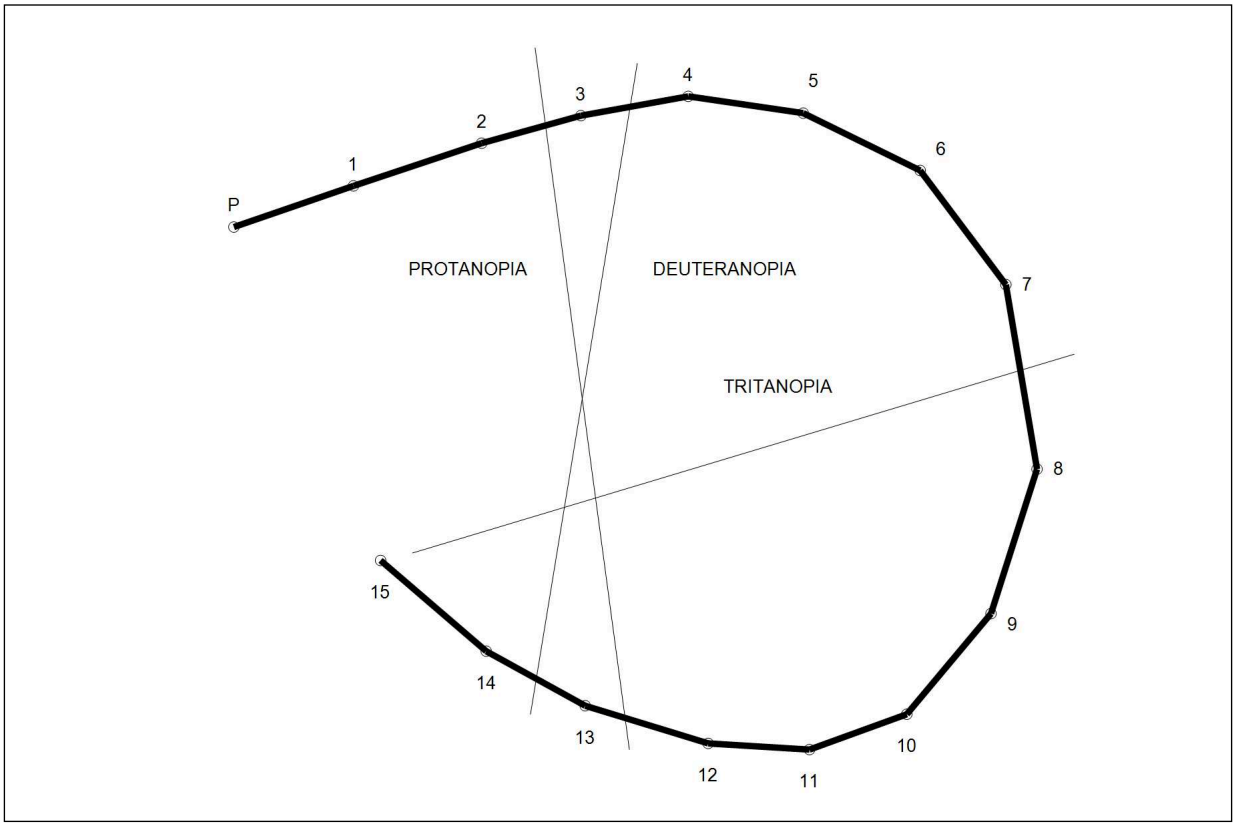
Seznam příloh

Pacient
Datum narození
Rodné číslo
Pojišťovna

FAKULTNÍ NEMOCNICE OSTRAVA

Test barvocitu - D15

Datum vyšetření	Typ testu	Test provedl
19.04.2017	Na počítači	Lukáš Kolarčík



Oko je smyslový orgán reagující na světlo (fotoreceptor), tedy zajišťující zrak. V průběhu vývoje živočichů došlo k výraznému rozvoji od světločivných orgánů reagujících pouze na existenci světla až po oko „jednoduché“ u obratlovců (včetně člověka) a hlavonožců a oko složené např. u členovců. Mnoho živočichů (většina savců, ptáků, plazů a některých ryb) má oči umístěné na přední straně hlavy, což jim umožňuje trojrozměrné binokulární vidění - jednotlivé obrázky z obou očí se skládají do jednoho, který vnímáme. Oproti tomu existuje monokulární vidění, kdy živočich vnímá současně dva rozdílné obrazy, jako např. zajíc nebo chameleon.

- Lukáš Granzer

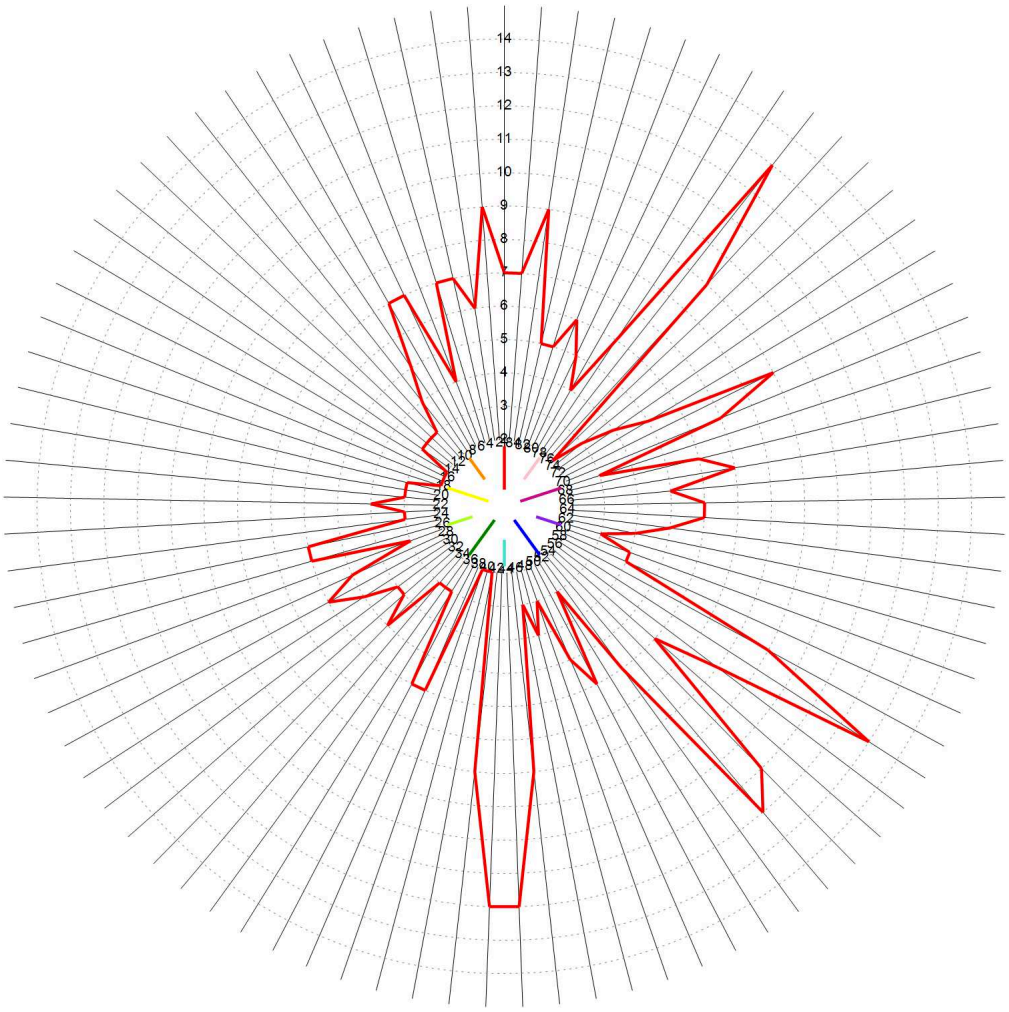
Příloha A - Export testu D15

Pacient
Datum narození
Rodné číslo
Pojišťovna

FAKULTNÍ NEMOCNICE OSTRAVA

Test barvocitu - Hue 100

Datum vyšetření	Typ testu	Test provedl
12.04.2017	Ruční zadání	Lukáš Kolarčík



Zde bude text pacientovi diagnózy.
Zpráva může být mít i několik odstavců. Na konec zprávy se automaticky přidává podpis lékaře, který prováděl změnu tohoto textu.

- Lukáš Granzer

Příloha B - Export Farnsworth-Munsellova testu

Pacient
Datum narození
Rodné číslo
Pojišťovna

Test ostrosti zraku

Datum vyšetření	Vzdálenost	Test provedl
22.02.2018	10 m	Michaela Šidiková

	Levé oko			Pravé oko		
	pokusů	správně	úspěšnost	pokusů	správně	úspěšnost
10 / 100	7	7	100%	7	7	100%
10 / 66	7	7	100%	7	7	100%
10 / 40	7	7	100%	7	7	100%
10 / 33	7	7	100%	7	7	100%
10 / 20	7	7	100%	7	7	100%
10 / 15	7	7	100%	6	0	0%
10 / 10	7	7	100%	-	-	... %
Celkem			100 %			83 %

Zde můžeme vidět, že pacient hůře vidí na pravé oko. Levé oko je zcela zdravé.
Na pravém oku můžeme pozorovat zhoršení vidění do dálky - krátkozrakost.

- Lukáš Granzer

Příloha C - Export testu ostrosti zraku